The apportionment problem

A monograph

by

Bjørne-Dyre Hougen Syversten

Dissertation submitted to the Norwegian School of Economics and Business Administration in partial fulfillment of the requirements for the degree of Doctor oeconomiae

ISBN: 82-405-0039-0

Contents

Part II

Appendices

Preface

This dissertation deals with the field of study known as the apportionment problem. In addition to giving an overview of the field, it treats some topics in greater detail. The dissertation is divided in two parts. The first part, chapter 1 through 8, deals with the vector apportionment problem, while the second part, chapter 9 through 18, treats the matrix apportionment problem. In addition to the text, the dissertation contains 4 appendices.

The apportionment problem concerns the problem of dividing the seats in an elected assembly fairly, i.e. proportionally, among the constituencies and/or parties according to the populations/votes. What makes it impossible to achieve a perfectly proportional division in every situation is the indivisibility of seats. During history many methods for apportioning seats have been proposed. The most important of these are presented in chapters 1 and 2. Due to the indivisibility of seats, it is not clear what one should put into the term proportional in the apportionment context. Chapter 3 reviews some natural principles in the vector case, while chapter 9 presents some principles in the matrix case. Although the common use of apportionment methods is distribution of seats in elected assemblies, it should be noted that such methods are applicable in every situation where indivisible entities are to be distributed proportionally. The test of time has revealed the properties of different apportionment methods. Chapter 4 studies some properties, while chapter 8 contains a discussion regarding suitable properties for different kinds of apportionment. Apportionment methods may also be formulated as constrained optimization problems. Chapter 6 presents formulations for the vector case and chapters 10 - 11 for the matrix case.

I first became aware of the apportionment problem when I read Balinski and Young's book "*Fair Representation: Meeting the Ideal of One Man, One Vote*" in the early 1990s. My mentor, Professor Kurt Jörnsten of the Norwegian School of Economics and Business Administration, then drew my attention to the matrix apportionment problem. An unpublished working paper he had written together with Professor Thorkell Helgason of the University of Iceland, which presented

an entropy formulation of the matrix problem, became the starting point for my HAS-thesis. During the work with that thesis I tried to find an efficient algorithm for solving the matrix apportionment problem. The thesis (1992) ended with a sketch of an apportionment algorithm.

The early part of the work with the doctoral dissertation focused on developing the apportionment algorithm. The process of programming the algorithm contributed to its development. Chapter 12 presents the proposed algorithm. Another task has been the testing of different set-ups of the algorithm. The results of these tests are presented in chapter 14. Chapter 17 is also connected to the algorithm. It presents a way of decomposing the multiplier set for an optimal matrix apportionment. An important part of the dissertation is the empirical measurement of bias for both the vector and matrix apportionment problem. In this connection I propose new ways of grouping the constituencies in chapter 5. The results of the bias tests are presented in chapters 5 and 16 respectively. Three other chapters worth mentioning are chapter 7 which deals with thresholds, chapter 15 which describes controlled rounding, and chapter 18 which presents the three-dimensional apportionment problem.

Compared with my HAS-thesis, the contents of chapters 5, 7, and 12 - 18 are new, while the contents of the other chapters have been refined. In a wider perspective, I consider the following to be the new contributions of the dissertation:

- The apportionment algorithm.
  [The apportionment initialization procedure (section 11.3) plus the rules for the adjustments (chapter 12)]
- A constructed measure for Norway.
  [Section 8.2]
- The measurement of bias of matrix apportionments.
  [Chapter 16]
- Division of constituencies (cells) in new ways when measuring bias.
  [Sections 5.3 - 5.6]
- Some minor results in part I regarding CP_t may also be characterized as new.
- The payoff functions in chapter 7 were derived independently of Lijphart and Gibberd. Compared with their article, only the general functions for constant parametric divisor methods represent something new.

During the last couple of years I have revised the dissertation. A lot of errors have been corrected during this revision. The contents of the language have also been improved. However, since my English could have been better, there is certainly some bits of incorrect English left.

The comments of Professor Thorkell Helgason of the University of Iceland have been valuable during the revision process. Thanks also to my mentor, Professor Kurt Jörnsten of the Norwegian School of Economics and Business Administration, for ideas and discussions along the way. Finally, thanks to Professor Aanund Hylland of the University of Oslo for his comments on my HAS-thesis.

I also wish to thank the Norwegian School of Economics and Business Administration and Sør-Trøndelag College, School of Economics and Business Administration for financial support during the lengthy work with the dissertation. Finally, thanks to the Norwegian School of Economics and Business Administration, Department of Finance and Management Science for provision of office after my employment period there.

Trondheim, December 1998 / October 1999

Bjørne-Dyre Hougen Syversten

Key terms

Notation

	Stands for	Introduced
Basic apportionment notation		
h	House size	Page 3
m	Number of constituencies	Page 3
n	Number of parties	Page 103
o	Number of levels	Page 281
i	Index for constituencies	Page 3
j	Index for parties	Page 103
k	Index for levels	Page 281
l	Index for seats	Page 11
M	The set of all constituencies	Page 3
N	The set of all parties	Page 103
O	The set of all levels	Page 281
H	The set of seats	Page 11
$\mathbf{p}$	Population vector / Vote matrix	Page 3, 139, 281
p_i	Population in constituency i	Page 3
p_{ij}	Votes for party j in constituency i	Page 139
p_{ijk}	Votes for level k of party j in constituency i	Page 281
$\mathbf{a}$	Apportionment vector/matrix	Page 4, 140, 281
a_i	Apportionment to constituency i	Page 4
a_{ij}	Apportionment to party j in constituency i	Page 140
a_{ijk}	Apportionment to level k of party j in constituency i	Page 281
A	Apportionment method	Page 4
$(\mathbf{p},h)$	Election situation for vector problem	Page 4
p	National average population per seat	Page 6
p_i	Average number of people per seat for constituency i	Page 168
$\mathbf{q}$	Quota vector/matrix	Page 6, 145
q_i	Quota of constituency i	Page 6
r_i	Fractional part of quota	Page 8
r^*	Smallest fraction which wins a seat with LF	Page 9
d_l	Divisor number l in a divisor series	Page 11
z	Common divisor	Page 18
z_J	Common divisor for HA	Page 19
z_A	Common divisor for SD	Page 19

$\hat{r}_i$	Remainder of constituency quotient	Page 50
t	Parameter for constant parametric divisor method	Page 22
$\mathbf{r}$	Lower constituency bounds vector	Page 141
$\mathbf{R}$	Upper constituency bounds vector	Page 141
r_i	Lower bound for constituency (row) i	Page 141
R_i	Upper bound for constituency (row) i	Page 141
$\mathbf{c}$	Lower party bounds vector	Page 142
$\mathbf{C}$	Upper party bounds vector	Page 142
c_j	Lower bound for party (column) j	Page 141
C_j	Upper bound for party (column) j	Page 141
σ	Bound vector	Page 143, 282
$(\mathbf{p},\sigma)$	Election situation for matrix problem	Page 143
$\mathbf{f}$	Allocation matrix	Page 143
$R(\sigma)$	Set of allocations	Page 143
$R^0(\mathbf{p},\sigma)$	Subset of $R(\sigma)$	Page 149
$R^+(\mathbf{p},\sigma)$	Subset of $R^0(\mathbf{p},\sigma)$	Page 149
$R^1(\mathbf{p},\sigma)$	Subset of $R^0(\mathbf{p},\sigma)$	Page 151
$\underline{I}$	Complement of the subset I	Page 149

Abbreviations

LF	The method of the largest fraction	Page 9
HA	The method of the highest average	Page 13
MF	The method of major fractions	Page 15
SD	The method of the smallest divisor	Page 17
EP	The method of equal proportions	Page 15
HM	The harmonic mean method	Page 16
DM	The Danish method	Page 22
CP_t	Constant parametric divisor method with parameter t	Page 22
EL	Election apportionment	Page 221
X-Y	Bound vector for which the constituency and party bounds have been determined by apportionment methods X and Y respectively	Page 221

Algorithm notation

δ	House size multiplier	Page 147
λ_i	Constituency multiplier for constituency i	Page 147, 159
μ_j	Party multiplier for party j	Page 147, 159
S	Set of positive vote cells	Page 155
$\overline{S}$	Set of zero vote cells	Page 155
a_{ijl}	0/1 variable for the lth seat in cell (i,j)	Page 156
$L(\lambda)$	Constituency relaxed objective function	Page 167
$\tilde{\mathbf{a}}$	Current assignment	Page 165
ρ	Measure of goodness	Page 165
ρ_i	Constituency i's contribution to ρ	Page 183
M^-	Set of underrepresented constituencies	Page 165
M^+	Set of overrepresented constituencies	Page 165
M^0	Set of rightly represented constituencies	Page 166
$\mathring{\lambda}_i$	Initial constituency multiplier	Page 168
t_i	Target quotient for constituency i	Page 170
τ	Target weight	Page 170
$\mathring{\kappa}$	Initial marginal value of representation	Page 170
$\mathring{\rho}$	Initial measure of goodness	Page 170
Q	Set of quotients which are assigned a seat	Page 184
$\overline{Q}$	Set of quotients which are unassigned	Page 184
q_{ijl}	Quotient number l in cell (i,j)	Page 184
$q^C_{uv}(x)$ ...	xth challenging quotient in cell (u,v)	Page 185
$q^O_{uv}(x)$	The xth outgoing quotient in cell (u,v)	Page 189
$q^B_{uv}(x)$	xth benchmark quotient in connection with cell (u,v)	Page 185, 190
$d^c_{ue}(x_e)$ ..	xth distance for constituency u within party e and where the benchmark quotient comes from constituency c	Page 195
$d^{ce}_u(y)$	yth global distance for constituency u, which occurs within party e and where the benchmark quotient comes from constituency c	Page 195
χ_u	Adjustment distance for constituency u	Page 187
υ	Adjustment weight	Page 187
$\overline{\lambda}_u$	Former value of constituency multiplier	Page 188
$\overline{q}_{ujl}$	Former value of quotient	Page 187
ϕ_i	ρ-effect for constituency i	Page 194
$\overline{M}$	The opposite malrepresentation group	Page 194
$\hat{\rho}_i$	Variable for remaining malrepresentation for constituency i	Page 194

Sundry

$\lfloor \zeta \rfloor$	Highest integer not larger than ζ	Page	8
$\lceil \zeta \rceil$	Lowest integer not smaller than ζ	Page	8
$u(l)$	Utility an individual derives from being represented by l representatives	Page	92
ε_{DE}	Bias between group D and E	Page	81
H_0	Null hypothesis	Page	82
H_A	Alternative hypothesis	Page	82
a	Miscellaneous		
s	Miscellaneous		

Controlled rounding

$\mathbf{b}$	A tabular array (matrix)	Page	237
b	Rounding base	Page	237
$\bar{a}_{ij}$	Integer part of entry in $\mathbf{b} \cdot \frac{1}{b}$	Page	239
$\tilde{b}_{ij}$	Fractional part of entry in $\mathbf{b} \cdot \frac{1}{b}$	Page	239
$\hat{\mathbf{b}}$	$\mathbf{b}$ transformed to canonical form	Page	239
$\hat{a}_{ij}$	0/1 rounding of $\tilde{b}_{ij}$	Page	240
z_p	Objective function for controlled rounding problem	Page	240

Grouping

c	Number of groups	Page	64
g	Index for groups	Page	64
C	The set of all groups	Page	64
G	The set of all constituencies belonging to g	Page	64
$\mathbf{n}$	Grouping vector	Page	64
n_g	Number of constituencies in group g	Page	64

Part I:

The Vector Apportionment Problem

Chapter 1: Introduction

Chapter 1 presents the vector apportionment problem. Section 1.1 introduces the basic terminology and presents some basic conditions, while the main topic in the last section is the apportionment method called the method of the largest fraction.

1.1 Basic terminology and conditions

To introduce the basic terminology for the vector apportionment problem we look at the situation where h seats shall be distributed among m constituencies based on the populations of these constituencies. We could just as well have looked at the apportionment of seats among parties because the methodology is similar. However, the apportionment among constituencies is chosen. $h = 0$ results in the trivial situation where no constituency gets any seat, while $m = 0$ means that there is not any constituency to distribute the seats to. We therefore let the **house size** h and the **number of constituencies** m be positive integers. If $m = 1$, the sole constituency must get all h seats available. Thus, for all practical purposes we can assume that $m \geq 2$. The index i, where $i \in M = \{1, 2, ..., m\}$ is used for the constituencies. The populations of the constituencies, or alternatively the number of eligible voters in the constituencies, are represented by the **population vector**, $\mathbf{p} = (p_i)$. A constituency with a population of zero is meaningless, so we assume that all populations are positive $\mathbf{p} > 0$. Usually p_i is integer, but when the result of an election is given as percentages, p_i is rational. For other areas where the use of apportionment methods might be desirable, p_i may be real. To cover all possibilities we assume that p_i is positive and real.

Population is just one possible basis for apportionment of seats in parliament. Other possible bases are the number of eligible voters, the number of votes, or another quantitative measure, including **constructed measures**. An example of a constructed measure is: Population + Eligible voters + 20·Area, which is used in Denmark, [H-S] (page III.4). In section 8.2 we propose a constructed measure for Norway.

The final apportionment of seats among the constituencies is represented by the **apportionment vector**, which we denote $\mathbf{a} = (a_i)$. A seat cannot be divided, so a_i must be integer valued. It is possible that constituency i is not apportioned any seat. Thus, a_i is a non-negative integer, i.e. $\mathbf{a} \geq 0$ and integer. It is obvious that the sum of seats over all constituencies a_M equals h:

$$\sum_{i \in M} a_i = a_M = h \tag{1.1}$$

We let p_M denote the total population of the country:

$$p_M = \sum_{i \in M} p_i \tag{1.2}$$

The vector $(p_1, p_2, \ldots, p_m, h)$, which we usually abbreviate to $(\mathbf{p},h)$, contains all data of interest for the apportionment. We call such a vector an **election situation** for the **vector apportionment problem**. The task ahead is to determine an apportionment vector which is "proportional" to the population vector. A method utilized in such a determination process is called **apportionment method** and denoted A.

An apportionment method should be able to handle every possible election situation, i.e. A should be well defined and non-empty for all election situations. It is reasonable to allow more than one possible apportionment for election situations where there is a tie between two or more constituencies for the last seat(s). A simple example:

Example 1.1
We face an election situation where h is an odd number and the country consists of two constituencies with equal populations. Then any reasonable apportionment method should allow both $\mathbf{a} = (\frac{h+1}{2}, \frac{h-1}{2})$ and $\mathbf{a} = (\frac{h-1}{2}, \frac{h+1}{2})$ as apportionments.

Ties can also arise when populations are unequal, but this depends on the properties of the apportionment method being used. Even if $\mathbf{p}$ is integer or rational, ties may involve irrational numbers. A tie can be broken by lottery, toss of coin or another tie breaking procedure. However, in most practical election situations the populations are so large that ties rarely occur.

A tie can be thought of as a point where the apportionment is about to change. In the immediate neighbourhood of a tie point arbitrarily small population changes will lead to different apportionments, and all these apportionments should be allowed at the tie point. The fact that a tie point together with its immediate neighbourhood involve irrational numbers is why we allowed $\mathbf{p}$ to be real from the beginning. [B&Y] (page 98) define a condition called **completeness**: Completeness is a continuity condition which extends the concept of apportionment methods to all real populations. An apportionment method A is completed by letting $\mathbf{a} \in A(\mathbf{p},h)$ for real population vectors $\mathbf{p} \in \Re^m$ if and only if there is a sequence of rational m-vectors $\mathbf{p}^a$ converging to $\mathbf{p}$ such that $\mathbf{a} \in A(\mathbf{p}^a,h)$ for all a. The mathematical formulation of completeness in [B&D-1] (page 711) is:

Definition 1.1
A is **complete** if $\mathbf{p}^a \to \mathbf{p}$ and $\mathbf{a} \in A(\mathbf{p}^a,h)$ for every a, then $\mathbf{a} \in A(\mathbf{p},h)$.

The central question regarding the apportionment problem is: When restricted to integer solutions, what do we mean by proportional? Below we review some basic conditions concerning proportionality.

Proportionality concerns the size of populations, not their names or other characteristics. Therefore, permuting the populations should only result in apportionments that are permuted in the same way. [H-A] (page 23) calls this condition **neutrality**:

Definition 1.2
A is **neutral** if for all h, $\mathbf{p}$, and $\mathbf{a}$ and all permutations ω of M,
$\mathbf{a} \in A(\mathbf{p},h)$ if and only if $\mathbf{a}_\omega \in A(\mathbf{p}_\omega,h)$.

Other names which have been used for this condition are **symmetry**, [B&Y] (page 97), and **anonymity**, [B] (page 139). An apportionment method which orders constituencies alphabetically and breaks ties in favour of the constituency ordered first is not neutral.

The same proportional change in the population of every constituency should not alter the apportionment, since there is no change in the proportional shares of the constituencies. [B&Y] (page 97) call this condition **homogeneity**, while [H-A] (page 9) uses the name **scale independence**.

Definition 1.3
A is **homogeneous** if $A(\mathbf{p},h) = A(\zeta\cdot\mathbf{p},h)$ for all $(\mathbf{p},h)$
and all real numbers $\zeta > 0$.

We use the notation p for the **national average population per seat**:

(1.3) $$p = \frac{p_M}{h}$$

The **quota** of constituency i is denoted q_i and defined as the population of this constituency divided by the national average population per seat:

(1.4) $$q_i = \frac{p_i}{p} = \frac{p_i}{p_M}\cdot h$$

q_i shows the exact number of seats constituency i would be entitled to if partial seats were allowed. The **quota vector** is denoted $\mathbf{q} = (q_i)$. It can be interpreted as the ideal assignment. Clearly, the sum of all quotas equals h:

(1.5) $$\sum_{i\in M} q_i = \sum_{i\in M} \frac{p_i}{p_M}\cdot h = \frac{h}{p_M}\cdot\sum_{i\in M} p_i = \frac{h}{p_M}\cdot p_M = h$$

In practice there are few election situations where any q_i is integer. When the quota of every constituency is integer, we have **exact proportionality**. In case this highly unlikely event occurs, the apportionment method should distribute the seats according to the quotas. This condition is called **exactness**, [B] (page 139), or **weak proportionality**, [B&Y] (page 97).

Definition 1.4
A is **exact** if $q_i \in \mathbb{N}$ for every i implies that $A(\mathbf{p},h) = \mathbf{q} = (q_i)$.

[B&Y] (page 97) and [B] (page 139) present the condition that as the house size grows the apportionment should become "no less proportional". Let us call this condition **integer proportionality**:

Definition 1.5
A is **integer proportional** if $\mathbf{a} \in A(\mathbf{p},h)$ and $\hat{\mathbf{a}} = \xi \cdot \mathbf{a}$ is integer valued, where $0 < \xi < 1$ and rational, imply that $\hat{\mathbf{a}} = A(\mathbf{p},\hat{h})$, where $\hat{h} = \xi \cdot h$.

Notice that $\hat{\mathbf{a}}$ should be the unique apportionment with the smaller house size $\hat{h}$. Consider an election situation with two constituencies and a house size of 6. If A apportions 4 seats to the first and 2 seats to the second constituency, then integer proportionality demands that A gives 2 seats to the first and 1 seat to the second constituency when the house size is only 3.

An apportionment method which is neutral, homogeneous, exact, and integer proportional is **proportional**, [B] (page 139). All methods we encounter in this and the next chapter are proportional and complete.

Whenever two constituencies have equal populations, it seems reasonable that their apportionments do not differ by more than one seat. The reason for allowing the apportionments to differ by one seat is that the two constituencies may have to share an odd number of seats, as in Example 1.1 above. Apportionment methods which obey the stated condition are called **balanced**, [B&Y] (page 144), or **strongly balanced**, [H-A] (page 24):

Definition 1.6
A is **balanced** if $\mathbf{a} \in A(\mathbf{p},h)$ and $p_i = p_s$ imply $|a_i - a_s| \leq 1$.

A natural requirement is that a constituency with a larger population than another constituency should never get less seats. This condition has been presented under the name weak population monotonicity, [B&Y] (page 147). [H-A] (page 27) calls it **internal vote monotonicity**:

Definition 1.7
A is **internal vote monotone** if $\mathbf{a} \in A(\mathbf{p},h)$ and $p_i > p_s$ imply $a_i \geq a_s$.

1.2 The method of the largest fraction

Let $\zeta \geq 0$ be a real number. We define $\lfloor \zeta \rfloor$ and $\lceil \zeta \rceil$ the following way:

(1.6) $\lfloor \zeta \rfloor$ = The greatest integer equal to or lower than ζ.
$\lceil \zeta \rceil$ = The smallest integer equal to or higher than ζ.

It is clear that $\lfloor \zeta \rfloor = \lceil \zeta \rceil$ if and only if ζ is integer valued. A constituency's quota can be divided in two parts, an integer and a fractional part. We denote the fractional part r_i, so the quota can be written as:

(1.7) $q_i = \lfloor q_i \rfloor + r_i$ where $0 \leq r_i < 1$

It seems reasonable that each constituency should get at least as many seats as the integer part of its quota $\lfloor q_i \rfloor$. After such an assignment there are still $\sum_{i \in M} r_i = r_M$ seats left for distribution. The number of remaining seats r_M is bounded the following way: $0 \leq r_M \leq m - 1$.

The normal way of dealing with fractions is to round fractions greater than $\frac{1}{2}$ upwards and fractions smaller than $\frac{1}{2}$ downwards, with $\frac{1}{2}$ as the tie point where fractions are rounded either upwards or downwards. We ignore the possibility of

fractions equal to $\frac{1}{2}$ in the following explanation: The described rounding procedure only works if the number of fractions greater than $\frac{1}{2}$ is equal to r_M. If there are more than r_M fractions greater than $\frac{1}{2}$, too many seats will be distributed, and if there are less, too few seats will be distributed.

The natural modification of the rounding procedure above is to sort the fractions in descending order and distribute the remaining r_M seats to the constituencies with the largest fractions. With this procedure the smallest fraction which qualifies for a seat will differ from one election situation to another. We denote such a "threshold" fraction r^*. For most election situations r^* is in the neighbourhood of 0,5. In the case of exact proportionality r^* is undefined.

The procedure described above is a well-known apportionment method. It is often called **the method of the largest remainder**. We prefer the name **the method of the largest fraction**, since this name is closest to our description of the method. The method is also known as **Hamilton's method**, named after Alexander Hamilton who proposed the method in 1792. Alexander Hamilton was the first Secretary of the Treasury of the United States and a prominent political figure in the early years of the US. We use the abbreviation **LF** for the method of the largest fraction. Algorithm 1.1 is a formal description of how the apportionment process is carried out with LF:

Algorithm 1.1

Step 1: Give each constituency as many seats as the integer part of its quota $\lfloor q_i \rfloor$.

Step 2: Order the fractional parts r_i in descending order. If there is a tie for a position, break it in favour of any of the eligible fractions. Find the sum of all fractional parts r_M, either as $r_M = h - \sum_{i \in M} \lfloor q_i \rfloor$ or $r_M = \sum_{i \in M} r_i$. Give one seat each to the first r_M constituencies on the ordered fraction list.

Step 3: The final apportionment to a constituency is the sum of seats given to it in step 1 and 2.

Another way of distributing the last r_M seats is to let the constituencies with the largest fraction-integer ratios $\frac{n_i}{\lfloor q_i \rfloor}$ get one seat each. This method was proposed by William Lowndes in 1822 and is known as **Lowndes' method**. The criterion for apportioning the last r_M seats seems rather peculiar, and the method has never been used in practice.

Lowndes' method as described above is incomplete because $\frac{n_i}{\lfloor q_i \rfloor}$ is undefined for $0 < q_i < 1$. A definition like $\frac{n_i}{\lfloor q_i \rfloor} > \frac{r_s}{\lfloor q_s \rfloor} \geq 1$ if $0 < q_s < q_i < 1$ would make the method complete.

Chapter 2: Divisor methods

Most of the common apportionment methods belong to the class of divisor methods. The definition in section 2.1 is a broad definition of divisor methods. In section 2.7 we present an alternative definition which is narrower and better suited for our purpose. Each of the five sections between section 2.1 and section 2.7 presents a specific divisor method. In section 2.8 we divide the divisor methods in four categories. The following section describes the category we call constant parametric divisor methods. In the last section we look at the relationship between the family of constant parametric divisor methods and three other apportionment methods in some practical election situations.

2.1 A definition of divisor methods

Let us first introduce the index $l \in H = \{1, 2, ..., h\}$, which is related to the seats.

Definition 2.1

Let d_l be a non-negative real number where $d_l \leq d_{(l+1)}$ for all l. Then $0 \leq d_1 \leq d_2 \leq ... \leq d_l \leq ... \leq d_h$ is a non-decreasing series of non-negative real numbers called divisors. If $0 \leq d_1 < d_2 < ... < d_l < ... < d_h$ we have a **strict divisor method.** $\frac{p_i}{d_l}$ is the lth quotient for constituency i. We permit dividing by 0 and define $\frac{p_i}{0} > \frac{p_s}{0}$ if $p_i > p_s$. Moreover, $\frac{p_s}{0} > \frac{p_i}{\zeta}$ where ζ is any positive real number. The divisor method apportionment can then be found recursively as follows:

$$A(\mathbf{p},h) = \{\mathbf{a} \mid \sum_{i \in M} a_i = h, \max_{s \in M} \frac{p_s}{d_{(a_s+1)}} \leq \min_{i \in M} \frac{p_i}{d_{a_i}}\} \tag{2.1}$$

The easiest implementation of (2.1) is Algorithm 2.1 below, where the comments in parentheses refer to the first time the step is executed.

Algorithm 2.1

Step 1: Calculate the first quotient $\frac{p_i}{d_1}$ for every constituency. The constituency with the largest first quotient, any tie broken arbitrarily, wins the first seat.

Step 2: If h seats have been distributed, the final apportionment has been found.

Step 3: Eliminate the quotient which won the last seat (seat no. 1) and calculate the next quotient (the second quotient $\frac{p_i}{d_2}$) for this constituency. The other constituencies retain their current quotients. The constituency which now has the largest quotient wins the next seat (seat no. 2). If there is a tie between two or more quotients for this position, give the seat to any of the eligible quotients. Go to step 2.

We denote the sth largest quotient overall by q_s. All divisor methods assign one seat to each of the h largest quotients. With the algorithm above one only has to calculate $m + h - 1$ quotients to apportion the h seats.

Multiplying all divisors in a series by a positive real factor ζ does not alter the divisor method, because the ordering of the quotients does not change. Two divisor methods are identical if all divisors in the two series are equal after a scaling:

(2.2) A and $\hat{A}$ are identical if $d_l = \zeta \cdot \hat{d}_l$ for all l

There are infinitely many divisor methods. Some methods of practical and theoretical importance are presented in the next sections.

2.2 The method of the highest average

The method of the highest average is but one of several names for this common divisor method. Its divisor series consists of all positive integers, where the lth divisor in the series is defined as:

(2.3) $$d_l = l$$

Constituency i competes for its lth seat with the quotient $\frac{p_i}{l}$. This quotient shows the average population behind each seat for constituency i if it wins its lth seat. This explains the name the method of the highest average, which we later abbreviate to **HA**. Another name for the method is **the method of the greatest divisor**. In Europe the method is known as **d'Hondt's method** after Victor d'Hondt, a Belgian lawyer, who proposed the method in 1878. Yet another name is **Jefferson's method**, due to Thomas Jefferson, the third president of the US, who proposed the method in 1791. His formulation of the apportionment procedure differs from that shown above, [B&Y] (page 18):

"For a given house size h, find a value for the common divisor z_J so that the integer parts of the constituency quotients $\frac{p_i}{z_J}$ sum to h."

In section 2.7 it is explained why this formulation yields the same method.

2.3 The method of major fractions

This method is also widely used. The divisor series consists of the positive odd numbers. Therefore, it is often called **the method of odd numbers**. The lth divisor is given as:

(2.4) $$d_l = 2 \cdot l - 1$$

Another name for the method is **Laguë's method** after the French mathematican André Sainte-Laguë, who proposed the method in 1910. In general elections in Norway a modified version of the method is used. The first divisor in this modified method is 1,4, while the rest of the divisors are as defined above. The purpose of the modification is to make it harder for small parties to win their first seat.

As mentioned at the end of section 2.1, we are allowed to multiply all divisors in a series by a positive real factor. When we multiply the series in (2.4) by $\frac{1}{2}$ the formula for the lth divisor becomes:

(2.5) $$d_l = l - \frac{1}{2}$$

Usually it is more convenient to have the series in this form, and from now on we assume that it is.

The method is also called **Webster's method**. Daniel Webster was a lawyer and politician, US Secretary of State, who as early as 1832 proposed the method. His formulation follows Jefferson's approach, but rounds the constituency quotients in a different way, [B&Y] (page 32):

"For a given house size h, find a value for the common divisor z_W so that the constituency quotients $\frac{p_i}{z_W}$ rounded in the normal way sum to h."

This rounding procedure explains the name **the method of major fractions**, later abbreviated to **MF**.

2.4 The method of equal proportions

This method is used in the US for the apportionment of seats in the House of Representatives among the states. Its divisor series consists of the geometric means of all successive pairs of non-negative integers. The lth divisor in the series is defined as:

(2.6) $$d_l = \sqrt{(l - 1) \cdot l}$$

Since $d_1 = 0$, all constituencies will get at least one seat as long as the number of constituencies does not exceed the number of seats to be apportioned, i.e. $m \leq h$. If there are fewer seats than constituencies, Definition 2.1 demands that the h largest constituencies get one seat each. The first divisor is $d_1 = 0$, while the next four divisors in the series are approximately: 1,41, 2,45, 3,46, and 4,47. We notice that the divisors approach the corresponding divisors for MF as l increases. However, they will never be equal to those divisors, since $\sqrt{(l - 1) \cdot l} < (l - \frac{1}{2})$. The method got its name from Edward V. Huntington, a professor of mechanics and mathematics, who was a strong supporter of the method. Another name for the method is **Hill's method** after Joseph A. Hill, a statistican, who proposed the method in 1911, [B&Y] (page 48):

"For a given house size h, give to each state (constituency) a number of seats so that no transfer of any one seat can reduce the percentage difference in representation between those states (constituencies)."

This procedure explains the name **the method of equal proportions**. For a further explanation of the procedure see section 6.9. Later the abbreviation **EP** is used to refer to the method.

2.5 The harmonic mean method

The divisor series for **the harmonic mean method** consists of the harmonic means of all successive pairs of non-negative integers. The formula for the *l*th divisor in the series is:

$$(2.7) \qquad d_l = \frac{(l - 1) \cdot l}{l - \frac{1}{2}}$$

The first divisor is $d_1 = 0$, while the next four divisors are approximately: 1,33, 2,40, 3,43, and 4,44. With the exception of d_1, all divisors are smaller than the corresponding divisor for EP. We use the abbreviation **HM** for the method. HM was proposed in 1832 by James Dean, a professor of astronomy and mathematics, and is also known as **Dean's method**, [B&Y] (page 30):

"For a given house size *h*, find a common divisor z_D so that the whole numbers which make the average population per seat for the states (constituencies) closest to z_D sum to *h*."

This translates into giving constituency *i* $a_i + 1$ seats if $\frac{p_i}{a_i} - z_D \geq z_D - \frac{p_i}{a_i + 1} \geq 0$ and a_i seats if $z_D - \frac{p_i}{a_i + 1} \geq \frac{p_i}{a_i} - z_D \geq 0$, with $\frac{p_i}{a_i} - z_D = z_D - \frac{p_i}{a_i + 1}$ as the tie point.

2.6 The method of the smallest divisor

This method was used for the apportionment of seats in the French Assemblée Nationale among the departements in 1986, [B&D-2] (page 205). The divisor series consists of all non-negative integers. The *l*th divisor is defined as:

(2.8) $$d_l = l - 1$$

Like the two preceding divisor methods, $d_1 = 0$. When $h > m$ the method can be described as follows: Give each constituency one seat and distribute the remaining $h - m$ seats with HA. We recall that one of the names for HA was the method of the greatest divisor. It should now be clear why **the method of the smallest divisor** is one of the names for the method presented here. We abbreviate this name to **SD**.

Yet another name for the method is **Adams' method**, due to John Quincy Adams, the sixth president of the US, who proposed the method in 1832. His formulation of the apportionment procedure was, [B&Y] (page 28):

"For a given house size h, find a value for the common divisor z_A so that the constituency quotients $\frac{p_i}{z_A}$ rounded up to the nearest integer sum to h."

While HA rounds all constituency quotients downwards to the nearest integer, SD rounds them upwards to the nearest integer. Thus, SD's rounding procedure is the mirror image of HA's.

2.7 An alternative formulation of divisor methods

Before we present the alternative divisor method formulation, let us formally define a term we have used in the preceding sections: A **constituency quotient** $\frac{p_i}{z}$ is the quotient one gets when dividing the population of a constituency by a **common divisor** z, where z is a positive real number. The common divisor z must not be confused with a divisor d_l in a divisor series.

Definition 2.2
We form a subclass of the series of strict increasing divisors defined in Definition 2.1 by imposing the condition that the divisor function is a strictly monotone real function defined on the positive integers $l \geq 1$ and satisfying:

(2.9) $$l - 1 \leq d_l \leq l \text{ and}$$
$$\text{not } d_l = l \text{ and } d_b = b - 1 \text{ for integers } l \geq 1 \text{ and } b > 1$$

A divisor rounding of the constituency quotient $\frac{p_i}{z}$ is defined by:

(2.10)
$$\left[\frac{p_i}{z}\right]_d = 0 \quad \text{if } \frac{p_i}{z} < d_1$$
$$= 0 \text{ or } 1 \quad \text{if } \frac{p_i}{z} = d_1$$
$$= l \quad \text{if } d_l < \frac{p_i}{z} < d_{(l+1)}$$
$$= l \text{ or } l + 1 \quad \text{if } \frac{p_i}{z} = d_{(l+1)}$$

With this definition, the divisors d_l are the tie points where constituency quotients $\frac{p_i}{z}$ are rounded either up or down. The divisors can be seen as sign posts; when a constituency passes the sign post d_l, it gets its lth seat. There is one or two sign posts in the interval $[l - 1, l]$. The usual situation is one, but when $d_l = l - 1$ or l, and $d_{(l+1)} = d_l + 1$, there are two. The last part of (2.9) makes sure that the divisor method is exact, [B] (footnote page 139). Definition 2.2 holds for methods with $d_1 = 0$ if $h \geq m$.

Definition 2.2 is narrower than Definition 2.1. Take $d_l = l^2$ as an example. According to Definition 2.1 this is a strict divisor method. However, it is impossible to find a positive real factor ζ which scales this divisor series such that the interval requirement in (2.9) is satisfied for every divisor in the series. Proof: $\zeta \cdot d_4 = \zeta \cdot 16 \geq 3$ and $\zeta \cdot d_6 = \zeta \cdot 36 \leq 6$ imply $\frac{3}{16} \leq \zeta \leq \frac{1}{6}$, which is a contradiction. From now on we assume that methods we refer to as **strict divisor methods** satisfy Definition 2.2. Moreover, we assume that $h \geq m$ if divisor methods with $d_1 = 0$ is used.

For a given election situation and divisor method we must find a value for the common divisor z which distributes exactly h seats. How can appropriate z values be determined? We go back to Algorithm 2.1 in the first section to answer this question: For each divisor method there are two quotients of special interest, namely the largest quotient which does not win a seat, i.e. the $(h + 1)$-th largest quotient overall $q_{(h+1)}$, and the smallest quotient which does win a seat, i.e. the hth largest quotient overall q_h. It can be shown that proper values for z will lie in the interval $\langle q_{(h+1)}, q_h]$, which with a_i denoting the final number of seats apportioned to a constituency i can be expressed as:

$$z \text{ interval} = \langle q_{(h+1)}, q_h] = \langle \max_{s \in M} \frac{p_s}{d_{(a_s+1)}}, \min_{i \in M} \frac{p_i}{d_{a_i}}] \tag{2.11}$$

In sections 2.2, 2.3, and 2.6 we have denoted the common divisors for HA, MF, and SD z_J, z_W, and z_A respectively after Jefferson's, Webster's, and Adams' formulations. Below we investigate the relationsships between the national average population per seat p and these common divisors. The starting point in all three investigations is z set equal to p, which means that the quotas and the constituency quotients are equal at the outset.

HA rounds all constituency quotients down to the nearest integer. When the quotas are rounded this way, too few seats are distributed. The only exception to this rule is the case of exact proportionality where no rounding is required since all quotas are integer valued. To distribute more seats we must decrease z_J until exactly h seats are apportioned. Since all constituency quotients are rounded

downwards, the sum of these quotients will never be smaller than the number of seats. In (2.12) we also utilize that the number of seats can be expressed as $\frac{p_M}{p}$.

$$\sum_{i \in M} \frac{p_i}{z_J} = \frac{p_M}{z_J} \geq h = \frac{p_M}{p} \tag{2.12}$$

Simple manipulation of this expression yields that z_J will never exceed the national average population per seat $z_J \leq p$, and with the possibility of $z_J = p$ only in the case of exact proportionality.

SD rounds all constituency quotients up to the nearest integer. When the quotas are rounded this way, too many seats are distributed. The exception to this rule is the case of exact proportionality. To distribute fewer seats z_A must be increased until exactly h seats are apportioned. Since all constituency quotients are rounded upwards, the sum of these quotients will never exceed the number of seats:

$$\sum_{i \in M} \frac{p_i}{z_A} = \frac{p_M}{z_A} \leq h = \frac{p_M}{p} \tag{2.13}$$

From (2.13) we conclude that z_A will never be smaller than the national average population per seat $z_A \geq p$, and with the possibility of $z_A = p$ only in the case of exact proportionality. This is the complete opposite of the relationship with HA.

MF rounds constituency quotients the normal way. When the quotas are rounded this way, we can distribute too many seats, the right number of seats, or too few seats. Hence, z_W may be greater than, equal to, or smaller than the national average population per seat. It depends on the election situation at hand.

2.8 Categories of strict divisor methods

All divisor series which satisfy Definition 2.2 have at least one divisor in each of the closed intervals $[l - 1, l]$, so a general expression of them is:

(2.14) $\quad d_l = (l - 1) + F(l) \quad$ where $0 \leq F(l) \leq 1$ and real

We have let the parameter function $F(l)$ be a real valued function dependent on the number of the divisor l, but it is also possible to let it depend on more than one parameter. $F(l)$ only needs to be defined for integer values of l, although the function itself may be continuous in l.

It is natural to divide the strict divisor methods in four categories dependent on the behaviour of $F(l)$. When $F(l)$ is differentiable this can be done via the derivative.

Category 1:	$F(l)$ is a constant (sequence in l):	$F(l+1) = F(l)$ for all l.
Category 2:	$F(l)$ is an increasing sequence in l:	$F(l+1) > F(l)$ for all l.
Category 3:	$F(l)$ is a decreasing sequence in l:	$F(l+1) < F(l)$ for all l.
Category 4:	$F(l)$ is an unregular sequence in l:	$F(l+1)$ >/=/< $F(l)$.

$F(l) = \frac{l}{h}$ falls into category 2, while $F(l) = \frac{h - l}{h}$ belongs to category 3. Modified Laguë has $F(1) = 0{,}7$, but $F(l) = 0{,}5$ for $l \geq 2$, so it belongs to category 4. Strict divisor methods belonging to category 1 are of special interest because they are tractable. We take a closer look at these methods in the next section.

2.9 The constant parametric divisor methods

[O] (page 201→) defines the "parametric divisor method", which is category 1 above with $F(l) = t$. We call methods in category 1 **constant parametric divisor methods**. The constant parametric divisor methods are a family of divisor methods dependent on the real parameter t and with divisor series defined as:

(2.15) $\quad d_l = (l - 1) + t \quad$ where $0 \leq t \leq 1$ is a constant

Notice that $t = 0$, $t = \frac{1}{2}$, and $t = 1$ correspond to SD, MF, and HA respectively. We use the abbreviation $\mathbf{CP}_t$ for the constant parametric divisor method with parameter t. (2.15) is similar to the divisor method defined in [H-A] (page 45), where $d_l = l + \frac{\zeta - 1}{2}$ and $-1 \leq \zeta \leq 1$ and real. This corresponds to $t = \frac{\zeta + 1}{2}$, so the definitions are equal.

In general elections in Denmark the divisor series $d_l = 3 \cdot (l - 1) + 1$ is used for the apportionment of the parties' supplementary seats among the constituencies, [H-S] (page III.13). Multiplying this series by $\frac{1}{3}$ results in the equivalent divisor series $d_l = (l - 1) + \frac{1}{3}$. We call this divisor method **the Danish method** and abbreviate it to **DM**. Clearly, DM and $CP_{\frac{1}{3}}$ are the same method.

HM and EP are not constant parametric divisor methods because their parameter functions $F(l)$ are increasing sequences in l. $F(l)$ for HM is $\frac{l - 1}{2 \cdot l - 1}$, while $F(l)$ is $\sqrt{(l - 1) \cdot l} - (l - 1)$ for EP. Both these parameter functions approach $\frac{1}{2}$ asymptotically when $l \to \infty$.

2.10 Approximative t values for non-CP$_t$ methods

Based upon the Japanese 1985 census, [O] (page 206→) investigates which constant parametric divisor methods, i.e. which t values, that result in the same apportionment as with LF, HM, and EP respectively for Japan's House of Representatives. We have carried out a nearly similar analysis based on 13 countries. Our intervals for Japan only partially coincide with Oyama's reported intervals, and he does not explain in detail how his intervals have been determined. [B] (page 143) also comments on Oyama's investigation.

Before we describe how our analysis has been carried out, let us explain why we find the analysis interesting: Three very common apportionment methods belong to the family of constant parametric divisor methods; with SD and HA as the two extreme points and MF as the middle point on the t scale. The determination of the approximative placing of non-CP$_t$ apportionment methods on the t scale makes the t scale a kind of common ground and facilitates the comparison of different apportionment methods.

The usual situation is that many different constant parametric divisor methods, corresponding to different t values, result in the same apportionment. All t values which give the same apportionment form a ***t* interval**. The t intervals of special interest to us are the ones which result in the same apportionment as one of the non-CP$_t$ apportionment methods investigated. These intervals differ from one election situation to the next. Our reported t intervals for each combination of country and non-CP$_t$ apportionment method are based on the whole election situation. To denote the apportionment with a specific apportionment method we use **a** with the abbreviated name of the method as topscript, so $\mathbf{a}^{\mathrm{LF}}$ is the apportionment with LF etc. Some places in the description below we let $\mathbf{a}^{A}$ stand for the apportionment with a non-CP$_t$ apportionment method.

The t interval for a particular non-CP_t apportionment method in a given election situation consists of the t values for which a constant parametric divisor method gives the same apportionment:

$$t \text{ interval} = \{ t \mid \mathbf{a}^{CP_t} = \mathbf{a}^A \text{ for } t \in [0, 1]\} \tag{2.16}$$

This is easier said than done. The first task is to determine the non-CP_t apportionment, such that we know which apportionment $\mathbf{a}^{CP_t}$ shall be equal to. Below we present a systematic search process for determining the t interval, but let us first introduce the procedure and terminology we use when comparing apportionments:

We start by ordering the constituencies in descending order based on population size. This ordering facilitates the comparison of apportionments. There are two apportionments, denoted $\mathbf{a}$ and $\hat{\mathbf{a}}$, which shall be compared. Our comparison procedure starts with the constituency at the top of the constituency list. As long as the apportionments for the actual constituency are equal, i.e. $a_i = \hat{a}_i$, we continue down the list. The largest constituency for which the apportionments differ, here denoted s, is used to classify the relationship between the two apportionments. $\mathbf{a}$ is of **lower order** than $\hat{\mathbf{a}}$ if $a_s < \hat{a}_s$ and of **higher order** if $a_s > \hat{a}_s$. If the end of the constituency list is reached without any apportionments being unequal, $\mathbf{a}$ and $\hat{\mathbf{a}}$ are identical. The described procedure can be used for comparing any two apportionments for the same election situation.

Our search for t intervals starts by determining the apportionments for the reference methods CP_0, $CP_{\frac{1}{2}}$, and CP_1. We have the following rules which guide our adjustment of t: When t is increased and this results in a transfer of a seat from one constituency to another, the constituency which loses the seat will always be smaller than the constituency which gains. If t is decreased and this results in a transfer of a seat, the opposite holds. Section 4.3 presents the theoretical background for these rules.

For every chosen t value we compare the actual CP_t apportionment with the non-CP_t apportionment. If the CP_t apportionment is of lower order than the non-CP_t apportionment, t must be increased, while t must be decreased if the CP_t apportionment is of higher order. If the CP_t and non-CP_t apportionments are equal, we have discovered a value in the t interval.

We try to find the $\mathbf{a}^{CP_t} = \mathbf{a}^A$ apportionment by adjusting t as described above. It is possible that such a CP_t apportionment does not exist, because LF is not a divisor method and HM and EP are not constant parametric divisor methods. If we have established that the $\mathbf{a}^{CP_t} = \mathbf{a}^A$ apportionment exists, the next step in the determination of the t interval is to find the two neighbouring CP_t apportionments, i.e. the CP_t apportionment of lower order just "below" the $\mathbf{a}^{CP_t} = \mathbf{a}^A$ apportionment and the CP_t apportionment of higher order just "above" the $\mathbf{a}^{CP_t} = \mathbf{a}^A$ apportionment. A necessary condition for two CP_t apportionments to be neighbours is that they differ for only two constituencies. However, this is not a sufficient condition as Example 2.1 below illustrates.

We now present a procedure for determining the t value for the tie point between two neighbouring CP_t apportionments: The first step is to identify the two constituencies for which the apportionment changes. We denote the largest of them s and the smallest i. They get $a_s \geq 1$ and $a_i \geq 1$ seats respectively with the CP_t apportionment of lowest order, i.e. the one with the lowest t. At the tie point between the two apportionments, s' quotient no. $a_s + 1$ and i's quotient no. a_i are equal:

$$(2.17) \qquad \frac{p_s}{d_{(a_s+1)}} = \frac{p_i}{d_{a_i}} \Leftrightarrow \frac{p_s}{(a_s + 1) - 1 + t} = \frac{p_i}{a_i - 1 + t}$$

We solve for $t > 0$ which yields:

$$(2.18) \qquad t = \frac{[p_i \cdot a_s] - [p_s \cdot (a_i - 1)]}{p_s - p_i}$$

To check whether the two CP_t apportionments which were the basis for the calculation in (2.18) really are neighbours, we do the following: Let ε be a very small positive real number and determine the CP_t apportionments for the parameter values $t - \varepsilon$ and $t + \varepsilon$. If these apportionments are equal to the two apportionments we started out with, the calculated t value is the dividing point between two neighbouring t intervals, which means that the procedure for determining a tie point t value is brought to its end. On the other hand, if $\mathbf{a}^{CP(t-\varepsilon)}$ and/or $\mathbf{a}^{CP(t+\varepsilon)}$ are different from the two CP_t apportionments we started out with, the calculated t value does not represent a tie point. This means that we have discovered one (or two) CP_t apportionments which lie between these two apportionments. With the help of the CP_t apportionment(s) found for $t - \varepsilon$ and $t + \varepsilon$ and the two original CP_t apportionments, we create new pairs of possible neighbouring CP_t apportionments. For each pair we go through the procedure for determining the t value for the possible tie point. These actions are repeated until all tie points of interest have been determined. A t interval consists of all t values between two neighbouring tie points, including the tie points. Below we present the t intervals with three decimals accuracy. Moreover, we round all lower bound t values upwards and all upper bound t values downwards. Example 2.1 illustrates the procedure for determining t intervals:

Example 2.1

We utilize Japanese census data from 1985, see [O] (page 208-210). For a subset consisting of 3 constituencies and 9 seats, the apportionments with $CP_{0,32}$ and $CP_{0,34}$ are as follows:

Table 2.1

Constituency	Population	$\mathbf{a}^{CP_{0,32}}$	$\mathbf{a}^{CP_{0,34}}$
Kanagawa-5	1068400	4	5
Mie-2	574838	3	3
Hyogo-5	329052	2	1

Apportionments for three Japanese constituencies with $CP_{0,32}$ and $CP_{0,34}$.

These apportionments only differ for two constituencies, so they might be neighbouring apportionments. The t value for the possible tie point between Kanagawa-5 and Hyogo-5 is found with the help of formula (2.18):

$$t = \frac{(329052 \cdot 4) - (1068400 \cdot 1)}{1068400 - 329052} \approx 0{,}335$$

In this example we operate with $\varepsilon = 0{,}001$. Both $t - \varepsilon = 0{,}334$ and $t + \varepsilon = 0{,}336$ result in the apportionment (5, 2, 2). This CP_t apportionment lies between the two we started out with. Moreover, (4, 3, 2) and (5, 3, 1) are possible neighbours to (5, 2, 2) on either side, so both (4, 3, 2) together with (5, 2, 2) and (5, 2, 2) together with (5, 3, 1) are pairs of possible neighbouring apportionments. There is a possible tie point within each of these two apportionment pairs. We calculate the t values for these points:

$$t = \frac{(574838 \cdot 4) - (1068400 \cdot 2)}{1068400 - 574838} \approx 0{,}329 \quad \text{and} \quad t = \frac{(329052 \cdot 2) - (574838 \cdot 1)}{574838 - 329052} \approx 0{,}339$$

The apportionments for $t - \varepsilon = 0{,}328$ and $t + \varepsilon = 0{,}330$ are (4, 3, 2) and (5, 2, 2) respectively. Furthermore, the apportionments for $t - \varepsilon = 0{,}338$ and $t + \varepsilon = 0{,}340$ are (5, 2, 2) and (5, 3, 1) respectively. Hence, both apportionment pairs consist of neighbouring CP_t apportionments. The t intervals for the three CP_t apportionments we have encountered in this example are:

Table 2.2

Constituency	Population	$t \in [0{,}302, 0{,}329]$	$t \in [0{,}330, 0{,}338]$	$t \in [0{,}339, 0{,}342]$
Kanagawa-5	1068400	4	5	5
Mie-2	574838	3	2	3
Hyogo-5	329052	2	2	1

CP_t apportionments for three t intervals.

Both the lowest and the highest t value in Table 2.2, 0,302 and 0,342 respectively, have been determined from the whole election situation for Japan. Between these t values the presented subset of three constituencies determines the t intervals for the whole Japanese election situation.

Let us take a closer look at what happens when a CP_t apportionment changes. Recall that when t increases, the constituency which wins a seat will always be larger than the one which loses the seat. In the first transition in Table 2.2, the largest constituency wins a seat from the medium sized constituency, and in the second transition the medium sized constituency wins a seat from the smallest constituency. Thus, a seat is transferred from the smallest to the largest constituency in two steps. Such transfers involving several steps is why a difference in apportionment for only two constituencies is not a sufficient condition for two CP_t apportionments to be neighbours. The steps can also take place in reverse order. For a transfer process involving three constituencies this means that the medium sized constituency first takes a seat from the small constituency and thereafter loses it to the large constituency. Transfer processes may also involve more than three constituencies.

It should now be clear how t intervals are determined. Our results for the 13 countries investigated are presented in Table 2.3:

Table 2.3

Country	Year	Based on	Seats	m	HM		EP		LF	
Japan	1985	Population	512	130	Ø	Ø	0,397	0,455	0,456	0,481
Thailand	1970	Population	219	71	0,262	0,352	Ø	Ø	0,498	0,606
Greece	1981	Population	300	56	0,377	0,380	0,421	0,486	0,528	0,577
USA	1990	Population	435	50	Ø	Ø	0,385	0,498	0,510	0,537
Sweden	1990	Voters	349	28	0,366	0,701	0,366	0,701	0,366	0,701
Switzerland	1990	Population	200	26	0,294	0,399	0,294	0,399	0,488	0,529
Norway	1989	Voters	165	19	0,436	0,610	0,436	0,610	0,436	0,610
Denmark	1990	Voters	175	17	0,375	0,573	0,375	0,573	0,328	0,374
Germany	1990	Voters	656	16	0,143	0,514	0,143	0,514	0,515	0,553
Finland	1991	Voters	200	15	0,000	0,550	0,000	0,550	0,000	0,550
Canada	1981	Population	282	12	Ø	Ø	Ø	Ø	0,530	0,622
Austria	1991	Population	183	9	0,078	0,673	0,078	0,673	0,078	0,673
Iceland	1987	Voters	63	8	0,092	0,474	0,092	0,474	0,475	0,686
			Average		0,242	0,523	0,272	0,539	0,401	0,577

t intervals for which the non-CP_t methods HM, EP, and LF give the same apportionment as CP_t.

Comments on the t intervals for HM, EP, and LF in Table 2.3: Equal intervals for the same election situation mean equal apportionment, while non-overlapping intervals mean different apportionments. If an interval includes 0, 0,5, or 1, the apportionment is the same as with SD, MF, or HA respectively. Thus, the apportionment with HM, EP, and LF for Finland is the same as with SD and MF. We notice that Greece is the only country where the HM, EP, and LF apportionments all are different and at the same time correspond to CP_t apportionments. Moreover, Denmark is the only country where the LF apportionment is of lower order than the HM and EP apportionments. Finally, the HM and EP apportionments are identical for 8 of the countries, while $\mathbf{a}^{HM} = \mathbf{a}^{EP} = \mathbf{a}^{LF}$ for 4 countries.

So far we have talked about t intervals for non-CP_t apportionment methods, see equation (2.16). As should be clear from the explanations above, it is meaningful to talk about the t interval for a particular CP_t apportionment method in a given election situation. For instance, the t interval for MF ($CP_{0,5}$) in the Finnish election situation from 1991 is [0,000, 0,550].

A few thoughts about the number of CP_t apportionments for a given election situation, i.e. how many different apportionments there are when t varies from 0 to 1: This number is presumably positively correlated with the number of constituencies m and with the house size h. Another important factor is supposedly the distribution of constituency sizes.

From Table 2.3 one gets the impression that the LF apportionment always has a coinciding CP_t apportionment. This is a fact:

Theorem 2.1
There is always a t value for which $\mathbf{a}^{LF} = \mathbf{a}^{CP_t}$.

Proof: Let us first consider the case of exact proportionality. Both LF and CP_t are exact apportionment methods, so any CP_t will give the same apportionment as LF in this case. We move on and consider an arbitrary election situation where exact proportionality is not present. Make the LF apportionment and identify the

smallest fraction which qualifies for a seat r^*. It follows from the description in section 1.2 that $r^* \in \langle 0, 1 \rangle$. Consider the common divisor equal to the national average population per seat, i.e. $z = p$. With this common divisor the constituency quotients $\frac{p_i}{z}$ are identical to the quotas $\frac{p_i}{p}$. The constant parametric divisor method with $t = r^*$ rounds these constituency quotients the same way LF rounds the quotas. CP_{r^*} distributes exactly h seats and any tie in the LF apportionment has a corresponding tie here. Thus, $\mathbf{a}^{CP_{r^*}}$ is the counterpart to $\mathbf{a}^{LF}$ in every election situation where exact proportionality is not present. All kinds of election situations have now been covered. It has been verified that there exists a coinciding CP_t apportionment in every election situation. This completes the proof of Theorem 2.1.

The result in Theorem 2.1 makes it interesting to take a look at r^* for the election situations in Table 2.3:

Table 2.4

Country	Japan	Thailand	Greece	USA	Sweden	Switzerland	Norway	Denmark	Germany	Finland	Canada	Austria	Iceland
r^*	0,477	0,573	0,552	0,532	0,443	0,493	0,475	0,368	0,547	0,506	0,576	0,615	0,636

Threshold fraction r^* with LF for the 13 countries in Table 2.3.

$\mathbf{a}^{HM}$ for Japan, the US, and Canada and $\mathbf{a}^{EP}$ for Thailand and Canada are represented by $\varnothing$ in Table 2.3. This means that they have not got a $\mathbf{a}^{CP_t}$ counterpart in these election situations, i.e. the non-existence of identical CP_t apportionments. We use EP and its first two divisors to explain why this happens: The $\mathbf{a}^{EP} \cap \mathbf{a}^{CP_t} = \varnothing$ situation may involve other divisors, but the first two divisors, which are separated by the longest distance, are used for illustrative purposes. EP guarantees each constituency, however small, at least one seat. Of the constant parametric divisor methods only CP_0 (SD) does the same. At the same time, EP's next divisor $d_2 = \sqrt{2}$ is well above 1. The distance between the first two divisors is $\sqrt{2} \approx 1{,}41$ for EP compared to 1 for any CP_t. Viewed alone, EP's second divisor indicates a constant parametric divisor method somewhere in the neighbourhood of $CP_{0,41}$. Election situations with constituency quotients close to both 0,00 and 1,41 are sometimes a too difficult task to handle for one constant parametric divisor method, with the result that $\mathbf{a}^{CP_t} \cap \mathbf{a}^{EP} = \varnothing$.

Example 2.2 illustrates how we discovered that $\mathbf{a}^{CP_t} \cap \mathbf{a}^{HM} = \varnothing$ for the Japanese election situation from 1985:

Example 2.2

The three constituencies in Table 2.5 form the subset which can be said to be responsible for the non-existence of a CP_t apportionment equal to the HM apportionment for the whole Japanese election situation:

Table 2.5

Constituency	Population	$\mathbf{a}^{CP_{0,33}}$	$\mathbf{a}^{HM}$	$\mathbf{a}^{CP_{0,35}}$
Tokyo-10	1556469	6	7	7
Hokkaido-3	574984	3	2	3
Hyogo-5	329052	2	2	1

Election situation where $\mathbf{a}^{CP_t} \cap \mathbf{a}^{HM} = \varnothing$.

It can be verified that $\mathbf{a}^{CP_{0,33}}$ and $\mathbf{a}^{CP_{0,35}}$ are neighbouring apportionments within the class of constant parametric divisor methods and that $t \approx 0{,}340$ is the tie point between them. The $CP_{0,33}$ apportionment is of lower order than the HM apportionment, which itself is of lower order than the $CP_{0,35}$ apportionment. Since the two CP_t apportionments are neighbours, there does not exist a CP_t apportionment equal to the HM apportionment for the election situation in Table 2.5.

HM gives Hokkaido-3 only 2 seats, while CP_t insists on giving it 3 seats. Not before $t > 0{,}929$ does CP_t reduce its apportionment to Hokkaido-3 to 2 seats, resulting in the apportionment (8, 2, 1). On the other hand, only when $t \leq 0{,}340$ does CP_t distribute 2 seats to Hyogo-5. These two inequalities are clearly incompatible. Discovering such a contradiction is the normal way of finding out that there does not exist a CP_t apportionment equal to the non-CP_t apportionment. For the HM apportionment in Table 2.5 to have a CP_t counterpart, Tokyo-10 should have won its seat from Hokkaido-3 and not from Hyogo-5 when t was increased from 0,33. The divisors $d_2 = 1\frac{1}{3}$ and $d_3 = 2\frac{2}{5}$ are the reasons why Hyogo-5 holds on to its second seat and Hokkaido-3 is not able to win its third seat with HM.

Chapter 3: Conditions

In chapter 1 we presented what we called basic apportionment conditions. This chapter introduces some additional conditions. First we examine conditions which deal with the behaviour of the apportionment method when populations and/or the house size change. These dynamic conditions are important because the populations do change between elections and the house size may change. The second type of conditions examined are related to the quota. These static conditions set limits to the discrepancy between the actual apportionment and the quota. Unfortunately, it turns out that some dynamic and static conditions are incompatible.

3.1 House monotonicity

When the number of seats in the parliament changes while the populations of the constituencies remain unchanged, the apportionment should change in the same direction as h. The name of this condition is **house monotonicity** or **membership monotonicity**, [H-A] (page 19):

Definition 3.1
A is **house monotone** if for all $(\mathbf{p},h)$: $\mathbf{a} \in A(\mathbf{p},h)$ implies that there exist
an apportionment $\hat{\mathbf{a}} \in A(\mathbf{p},h+1)$ where $\hat{a}_i \geq a_i$ for all i and
an apportionment $\tilde{\mathbf{a}} \in A(\mathbf{p},h-1)$ where $\tilde{a}_i \leq a_i$ for all i.

When the house size is increased by one, Definition 3.1 requires that it is possible to give the extra seat to one of the constituencies and leave the representation of the other constituencies unchanged. The last part of the definition takes care of

the situation where the house size is decreased by one. The possibility of ties means that it is unreasonable to require that $\hat{\mathbf{a}}$ and $\tilde{\mathbf{a}}$ are the only apportionments in these situations. [B&Y] (page 117) and [B] (page 140) include only the first part of Definition 3.1 in their definitions of house monotonicity.

Theorem 3.1
Divisor methods are house monotone.

[H-A] (page 25) proves that a class of apportionment methods, which includes divisor methods, is house monotone. Let us present an intuitive explanation of Theorem 3.1: First we consider election situations without ties. Divisor methods distribute one seat at a time. It is meaningful to say that constituency i won the lth seat in the parliament. If the lth seat is given out or taken back, it only has implications for i's apportionment. Thus, divisor methods are house monotone. This explanation holds for tie situations too, but then we must keep track of how the ties are broken for different house sizes.

LF is not house monotone; a fact brought to light in the 1882 apportionment of the House of Representatives in the US. Then the state of Alabama would have got 8 seats with a house size of 299, but only 7 seats if the house size was increased to 300. This paradox has been known as the **Alabama paradox**. Example 3.1 in the next section illustrates the paradox.

3.2 Consistency

An apportionment which is acceptable to all constituencies should also be acceptable if restricted to any subset of constituencies considered alone. Thus, the way in which a group of constituencies share a given number of seats should be independent of the populations of the constituencies outside this group. This condition is called **uniformity**, [B&Y] (page 141), or **consistency**, [H-A] (page 17-18) and [B] (page 140). Consistency is an important condition because a constituency

will always compare its representation with other constituencies, especially those which fared better than itself.

The following notation is needed below: $\mathbf{k}_I$ is the subvector and h_I the number of seats for the constituencies in the subset $I \subset M$.

Definition 3.2
A is **consistent** if $\mathbf{a} \in A(\mathbf{p},h)$ implies that $\mathbf{a}_I \in A(\mathbf{p}_I,h_I)$;
and if $\hat{\mathbf{a}}_I$ is another apportionment for the subproblem,
then $\hat{\mathbf{a}}$ defined to be equal to $\hat{\mathbf{a}}_I$ on I and $\mathbf{a}$ elsewhere
is also an apportionment: $\hat{\mathbf{a}} \in A(\mathbf{p},h)$.

The part of the definition involving $\hat{\mathbf{a}}$ is necessary to handle tie points. It says that if a subproblem admits a tie, then there is a corresponding tie in the entire problem. In the definition of consistency in [H-A] (page 17-18) the set I consists of only two constituencies. By utilizing membership monotonicity and consistency, [H-A] (page 20-23) generalizes consistency to situations with more than two constituencies.

Theorem 3.2
Divisor methods are consistent.

Divisor methods are consistent because the seats are distributed one at a time such that the min-max inequality: $\max_{s \in M} \frac{p_s}{d_{(a_s+1)}} \leq \min_{i \in M} \frac{p_i}{d_{a_i}}$ is always satisfied, see [B&Y] (page 142). It makes no difference if i and s are members of a more restricted subset, because the ordering of their quotients is the same.

LF is neither consistent nor house monotone, as Example 3.1 illustrates:

Example 3.1
We utilize data from the general election in Sweden in 1991. The Riksdag has 349 seats, of which 39 are supplementary seats. The 310 fixed seats are apportioned among the 28 constituencies based on the number of eligible voters in the constituencies. LF is used for this apportionment, [H-S] (page III.18). In the

following we focus on the subset consisting of Stockholm county, Gothenburg municipality, and Älvsborg county South, which data are presented in Table 3.1. Together these three constituencies had 17,61% of the country's eligible voters and a combined quota of 54,60. We have calculated the quota for each constituency for a house size of both 54 and 55 seats. Table 3.1 shows the quotas and LF apportionments for these election situations. We notice that Älvsborg county South gets 7 seats when 54 seats are apportioned, but only 6 seats when 55 seats are apportioned. Hence, house monotonicity is violated.

Table 3.1

		h = 54		*h = 55*	
Constituency	Eligible voters	Quota	Seats	Quota	Seats
Stockholm county	672265	32,153	32	32,749	33
Gothenburg municipality	321822	15,392	15	15,677	16
Älvsborg county South	134949	6,454	7	6,574	6
Total	1129036	53,999	54	55,000	55

Älvsborg county South loses a seat when the total number of seats is increased from 54 to 55.

Gothenburg municipality and Älvsborg county South are geographical neighbours which presumably are interested in comparing their apportionments. Their quotas and LF apportionment for the election situation with 54 seats in Table 3.1 are repeated under the heading *with Stockholm* in Table 3.2. They get 15 and 7 seats respectively in this election situation. We are not interested in Stockholm county anymore and exclude its influence by calculating the quotas for Gothenburg municipality and Älvsborg county South based on the 22 seats they win. The quotas calculated and the resulting LF apportionment are shown under the heading *without Stockholm* in Table 3.2. The interesting point is that Älvsborg county South now only gets 6 seats, while Gothenburg municipality increases its representation to 16 seats. This is a violation of the consistency condition since LF gives a different apportionment to these two constituencies here than it did when they were a part of the three constituency subset.

Table 3.2

		with Stockholm		*without Stockholm*	
Constituency	Eligible voters	Quota	Seats	Quota	Seats
Gothenburg municipality	321822	15,392	15	15,5003	16
Älvsborg county South	134949	6,454	7	6,4997	6
Total	456771	21,846	22	22,0000	22

Quotas and LF apportionment for two election situations, of which the rightmost is a subset of the other.

The actual election outcome was that Stockholm county got 32, Gothenburg municipality 16, and Älvsborg county South 7 seats. From this information and the data in Table 3.1 we can conclude that there is another violation of the consistency condition, because the three constituencies divide the 55 seats differently in these two situations. Compared with the election situation where all 28 constituencies are taken into account, Stockholm county wins one seat from Älvsborg county South when restricting the attention to the election situation with $h = 55$ in Table 3.1.

3.3 Population monotonicity

Shifts in populations should be reflected in the apportionment. If the population of a constituency grows while the populations of all other constituencies remain the same, the constituency which population grows should get at least as many seats as before. **Weak population monotonicity** reflects this idea:

Definition 3.3
A is **weakly population monotone** if $\mathbf{a} \in A(\mathbf{p},h)$ and $\hat{\mathbf{a}} \in A(\hat{\mathbf{p}},h)$ with $\hat{p}_i > p_i$ and $\hat{p}_s = p_s$ for all $s \neq i$ imply $\hat{a}_i \geq a_i$.

Definition 3.3 is used in [H-A] (page 27) and [B] (page 140). In the former it is called **external vote monotonicity**. [B&Y] (page 147) use the name weak population monotonicity for the condition we have called internal vote monoton-

icity. There is a difference between internal and external vote monotonicity as explained by [H-A] (page 27→). The only thing Definition 3.3 says about the apportionments for the constituencies whose populations remain the same is that $\sum_{s \neq i} \hat{a}_s \leq \sum_{s \neq i} a_s$, so the new apportionment for a particular constituency in this group $\hat{a}_s$ may be smaller than, equal to, or greater than a_s. All apportionment methods described earlier are weakly population monotone.

Other definitions regarding how an apportionment method should behave when populations shift are conceivable; [B&Y] (page 117) use a definition which looks at the relative change in populations between two constituencies and call it population monotonicity. With this definition, [B&Y] (page 108-117) prove that a method is population monotone if and only if it is a divisor method. They are also able to prove that population monotonicity implies consistency and house monotonicity.

[B] (page 140) defines **population monotonicity** a little bit differently:

Definition 3.4
A is **population monotone** if $\mathbf{a} \in A(\mathbf{p},h)$ and $\hat{\mathbf{a}} \in A(\hat{\mathbf{p}},h)$
with $\hat{p}_i > p_i$ and $p_s > \hat{p}_s$ imply not $\{a_i > \hat{a}_i$ and $\hat{a}_s > a_s\}$.

Definition 3.4 with words: No constituency which gains in population should give up seats to a constituency which loses population.

3.4 Quota related conditions

It is reasonable that the actual apportionment is close to the quotas. The conditions presented in this section are different ways of measuring this. We start with the conditions **staying above lower quota** and **staying below upper quota**:

Definition 3.5
A **stays above lower quota** if $\lfloor q_i \rfloor \leq a_i$ for all i whenever $\mathbf{a} \in (\mathbf{p},h)$.

Definition 3.6
A **stays below upper quota** if $\lceil q_i \rceil \geq a_i$ for all i whenever $\mathbf{a} \in (\mathbf{p},h)$.

In [H-A] (page 15) these conditions are called **lower bound condition** and **upper bound condition** respectively. Theorem 3.3 and Theorem 3.4 below are given as a proposition in [B&Y] (page 130):

Theorem 3.3
HA is the unique divisor method which stays above lower quota.

Theorem 3.4
SD is the unique divisor method which stays below upper quota.

To prove that HA and SD stay above lower quota and below upper quota respectively, we go back to the alternative divisor formulation in section 2.7. In that section we found out that the common divisor for HA never is greater than the national average population per seat $z_J \leq p$, while the opposite $z_A \geq p$ is true for SD. By combining these relationships with the expression for the constituency quotient $\frac{p_i}{z}$, we get the relationship $\frac{p_i}{z_J} \geq \frac{p_i}{p}$ for HA and $\frac{p_i}{z_A} \leq \frac{p_i}{p}$ for SD. The right hand side of these inequalities is nothing but the quota $q_i = \frac{p_i}{p}$. Since the constituency quotients for HA are rounded downwards to the nearest integer to determine the apportionment while the constituency quotients for SD are rounded upwards, we have $a_i = \lfloor \frac{p_i}{z_J} \rfloor \geq \lfloor q_i \rfloor$ for HA and $a_i = \lceil \frac{p_i}{z_A} \rceil \leq \lceil q_i \rceil$ for SD. This proves that HA stays above lower quota and SD stays below upper quota.

To prove the uniqueness part of Theorem 3.3 we construct an election situation where only HA stays above lower quota. This is done as follows: From Definition 2.2 in section 2.7 we know that a divisor d_l lies in the interval $[l - 1, l]$. All divisors in the divisor series for HA are at the upper bound of such an interval because $d_l = l$. Thus, a divisor method which is different from HA must have at least one divisor which is lower than the corresponding divisor for HA. We

assume that only divisor number l is different, i.e. $d_l = l - \varepsilon$ where $1 > \varepsilon > 0$ is a real number. Suppose that this divisor method stays above lower quota. Then consider the following election situation: As usual we assume that $m \geq 2$. Constituency i competes for s seats, where $1 \leq s \neq l$, while the remaining constituencies compete for l seats each. We assume that these $m - 1$ constituencies are identical with a quota of $\frac{l\cdot(m - 1) - 1}{m - 1}$ each, i.e. a quota (just) below the integer l. The quota of constituency i is therefore:

$$q_i = h - (m - 1)\cdot\frac{l\cdot(m - 1) - 1}{m - 1} = h - l\cdot(m - 1) + 1$$

q_i is integer so staying above lower quota is satisfied when $a_i \geq h - l\cdot(m - 1) + 1$. This happens when i's quotient number $s = h - l\cdot(m - 1) + 1$ is higher than quotient number l for one of the other parties:

$$\frac{q_i}{d_s} = \frac{h - l\cdot(m - 1) + 1}{d_{[h - l\cdot(m - 1) + 1]}} > \frac{\frac{l\cdot(m - 1) - 1}{m - 1}}{d_l} \Leftrightarrow \frac{h - l\cdot(m - 1) + 1}{h - l\cdot(m - 1) + 1} > \frac{\frac{l\cdot(m - 1) - 1}{m - 1}}{l - \varepsilon} \Leftrightarrow$$

$$(l - \varepsilon)\cdot(m - 1) > l\cdot(m - 1) - 1 \Leftrightarrow \varepsilon\cdot(m - 1) < 1 \Leftrightarrow$$

(3.1) $$\varepsilon < \frac{1}{m - 1}$$

When $m \to \infty$, the right hand side of (3.1) approaches zero. Since ε must be lower than the right hand side, only $\varepsilon = 0$ suffices, which further means that only $d_l = l$ guarantees that the staying above lower quota condition is not violated. Thus, (3.1) contradicts the assumption that the divisor method here is different from HA. It is clear that when more than one divisor differs from HA's divisor series, election situations where staying above lower quota is violated can be constructed similarly. Hence, there exists no other divisor method than HA which stays above lower quota for all possible election situations.

The proof of that SD is the only divisor method which stays below upper quota follows a similar path as the proof for HA. Again we assume that only divisor number l is different from the original divisor series, i.e. $d_l = l - 1 + \varepsilon$. The quota of each of the $m - 1$ identical constituencies is $\frac{(l - 1)\cdot(m - 1) + 1}{m - 1}$, i.e. (just) above the integer $l - 1$. Constituency i's quota is therefore $q_i = h - [(l - 1)\cdot(m - 1) + 1]$,

i.e. an integer. The apportionment to i must be $a_i \leq h - (l - 1)\cdot(m - 1) - 1$ to stay below upper quota, and this happens when:

$$\frac{h - (l - 1)\cdot(m - 1) - 1}{d_{[h - (l - 1)\cdot(m - 1)]}} < \frac{\frac{(l - 1)\cdot(m - 1) + 1}{m - 1}}{d_l} \Leftrightarrow 1 < \frac{(l - 1)\cdot(m - 1) + 1}{(m - 1)\cdot(l - 1 + \varepsilon)} \Leftrightarrow$$

$$\varepsilon < \frac{1}{m - 1} \tag{3.2}$$

(3.2) is exactly the same condition as (3.1). When the number of constituencies m increases, we see that only $d_l = l - 1$ guarantees that the staying below upper quota condition is not violated. Thus, SD is the unique divisor method which stays below upper quota for all possible election situations.

Combination of Definition 3.5 and Definition 3.6 yields the stronger condition **staying within the quota**:

Definition 3.7
A **stays within the quota** if $\lfloor q_i \rfloor \leq a_i \leq \lceil q_i \rceil$ for all i whenever $\mathbf{a} \in A(\mathbf{p},h)$.

An equivalent formulation of staying within the quota is to require that the actual apportionment for every constituency is within one seat of the quota:

$$|q_i - a_i| < 1 \quad \text{for all } i \text{ whenever } \mathbf{a} \in A(\mathbf{p},h). \tag{3.3}$$

It is clear from the description of LF and Lowndes' method that they both stay within the quota. There is another story for the divisor methods:

Theorem 3.5
There is no divisor method which stays within the quota.

[H-A] (page 47-48) and [B&Y] (page 129-130) prove this theorem. In addition, they present the following results: All divisor methods stay within the quota when there are only two constituencies and MF even does so when the number of constituencies is limited to three. With four constituencies MF does not stay within the quota, as Example 3.2 below illustrates.

Let us present an intuitive explanation of Theorem 3.5: HA is the unique divisor method which stays above lower quota while SD is the unique divisor method which stays below upper quota. Since these two methods are different, there cannot exist a divisor method which stays within the quota.

Example 3.2

Table 3.3 shows all relevant data for the election situation:

Table 3.3

Constituency	A	B	C	D	Total
Population	241	74	53	32	400
Quota	12,05	3,70	2,65	1,60	20
LF apportionment	12	4	3	1	20
MF's divisor no.	12	4	3	2	
and its associated quotient	$10\frac{11}{23}$	$10\frac{4}{7}$	$10\frac{3}{5}$	$10\frac{2}{3}$	
MF apportionment	11	4	3	2	20

MF violates staying above lower quota and thereby staying within the quota.

Let us step into the MF apportionment process after 17 seats have been distributed. Then A, B, C, and D have got 11, 3, 2, and 1 seat respectively. This means that they compete for their next seat with their 12th, 4th, 3rd, and 2nd quotient respectively, which values are shown in the second row from the bottom in Table 3.3. The quotients for the three latter constituencies are greater than the quotient for A, which mean that the three remaining seats are given to D, C, and B, in that order. A's quota is greater than 12, but A does not get more than 11 seats, so the staying above lower quota condition is violated. It is also possible to construct examples where MF violates the staying below upper quota condition.

Staying within the quota looks at the absolute deviation between apportionment and quota. In relative terms this gives less leeway for the representation of a constituency with a large population than for one with a small population. Let us illustrate this by data from Example 3.2: A's apportionment is allowed to be at

most $\frac{0,05}{12,05} \approx 0,4\%$ smaller than or $\frac{0,95}{12,05} \approx 7,9\%$ greater than its quota. On the other hand, D's apportionment can be $\frac{0,6}{1,6} = 37,5\%$ smaller than its quota without violating the condition. Proportionality is about relative sizes. [B&Y] (page 80) therefore conclude that staying within the quota is not a reasonable condition for proportional apportionment methods.

There exist methods which are house monotone and stay within the quota, but they are not consistent, see [B] (page 141) and [B&Y] (page 134→). The reason why we have to choose between staying within the quota and consistency is the following impossibility theorem:

Theorem 3.6
No consistent apportionment method stays within the quota for all election situations.

[H-A] (page 47-48) proves the theorem. Theorem 3.6 is the main lesson from this chapter. Two important apportionment conditions are in conflict with each other.

The condition below is a digression in connection with the exactness condition from section 1.1. **Zero restrictedness** requires that whenever the quota of a constituency is integer, this is the apportionment for the constituency:

Definition 3.8
A is **zero restricted** if $q_i \in \mathbb{N}$ implies $a_i = q_i$ whenever $\mathbf{a} \in A(\mathbf{p},h)$.

We have borrowed the name zero restrictedness from a similar condition for the controlled rounding problem, see chapter 15. A more appropriate name in the apportionment context might be partial exactness. From the description in section 1.2 we draw the conclusion that both LF and Lowndes' method are zero restricted. None of the divisor methods is zero restricted. To illustrate this we manipulate Example 3.2 a little: Let A's quota be 12,00 and distribute the remaining 0,05 to B, C, or D. Clearly, A still would not get more than 11 seats.

Another condition which utilizes the quota is **staying near quota**, see [B&Y] (page 132), or **relative well roundedness** as [H-A] (page 41) calls it. A method satisfies this condition if it is impossible to take a seat from one constituency and give it to another and simultaneously bring both constituencies nearer to their quotas on a percentage basis:

Definition 3.9
A **stays near quota** if for all $\mathbf{a} \in A(\mathbf{p},h)$
there are no pair of constituencies $i \neq s$
such that $1 - \frac{a_i - 1}{q_i} < \frac{a_i}{q_i} - 1$ and $\frac{a_s + 1}{q_s} - 1 < 1 - \frac{a_s}{q_s}$.

Definition 3.9 is staying near quota in relative terms. The formulation in absolute terms is equivalent. In that case the requirements are: Not $\{q_i - (a_i - 1) < a_i - q_i$ and $a_s + 1 - q_s < q_s - a_s\}$. By dividing these expressions by q_i and q_s respectively, we arrive at the requirements in relative terms.

Theorem 3.7
LF and MF stay near quota. MF is the only divisor method which does so.

We start by establishing that LF stays near quota: Rearrangement of the requirements in absolute terms gives $\frac{1}{2} < a_i - q_i$ and $\frac{1}{2} < q_s - a_s$. For both these requirements to be satisfied, i's fraction $r_i = q_i - (a_i - 1)$, which is smaller than $\frac{1}{2}$, must be rounded up at the same time as s' fraction $r_s = q_s - a_s$, which is greater than $\frac{1}{2}$, is rounded down. This is impossible with LF, which therefore stays near quota. However, with Lowndes' method these roundings are possible. Consider the election situation with $\mathbf{q} = (10\frac{4}{5}, 1\frac{1}{5})$. The apportionment with Lowndes' method is $\mathbf{a} = (10, 2)$, which clearly violates staying near quota. Thus, staying within the quota does not imply staying near quota or vice versa.

[H-A] (page 42) and [B&Y] (page 132-133) prove the part of Theorem 3.7 regarding MF. That MF stays near quota is proved by assuming that an apportionment **a** does not stay near quota. Manipulation of the requirements yields that **a** then has to satisfy: $\frac{q_i}{a_i - \frac{1}{2}} < 1$ and $\frac{q_s}{a_s + \frac{1}{2}} > 1$ for some $i \neq s$. This violates the min-max inequality for MF. Thus, **a** cannot be an apportionment for

MF, which implies that MF stays near quota. That MF is the only divisor method which stays near quota is proved by showing that all other divisor methods violate staying near quota for some election situation.

Chapter 4: Properties

The first section of this chapter looks at how apportionment methods treat merger and division (of constituencies). These properties are of special interest when seats are distributed among political parties, because the political landscape is in continuous change. In section 4.2 we test assumptions regarding the distribution of remainders of constituency quotients and fractional parts of quotas. Closely related to merger and division is the treatment of small versus large constituencies. The theoretical part of this issue is dealt with in the last section, while the empirical side is the topic in the next chapter.

4.1 Treatment of merger and division

In this section we present four definitions concerning the behaviour of apportionment methods when constituencies merge or divide. We let $\mathbf{p}$ be the population vector before the merger (after the division) and $\hat{\mathbf{p}}$ the population vector after the merger (prior to the division). $\mathbf{p}$ and $\hat{\mathbf{p}}$ are equal with the exception of the elements p_i and p_s in $\mathbf{p}$, which are replaced with the element $\hat{p}_i = p_i + p_s$ in $\hat{\mathbf{p}}$. Thus, the former vector includes one more constituency than the latter, but the total populations $\mathbf{p}_M$ and $\hat{\mathbf{p}}_M$ are the same. [H-A] (page 66) presents Definition 4.1:

Definition 4.1
A **encourages merger** if $\mathbf{a} \in A(\mathbf{p},h)$ implies the existence of
$\hat{\mathbf{a}} \in A(\hat{\mathbf{p}},h)$ with $\hat{a}_{(i+s)} \geq a_i + a_s$ for all $i \neq s$ and all $(\mathbf{p},h)$.

An apportionment method A encourages merger if there always exists an apportionment where the merged constituency does not lose compared with the election situation before the merger.

Definition 4.2
A **encourages division** if $\hat{\mathbf{a}} \in A(\hat{\mathbf{p}},h)$ implies the existence of $\mathbf{a} \in A(\mathbf{p},h)$ with $\hat{a}_{(i+s)} \leq a_i + a_s$ for all $i \neq s$ and all $(\hat{\mathbf{p}},h)$.

An apportionment method A encourages division if there always exists an apportionment where the two new constituencies do not lose compared with the election situation before the division.

The word "existence" is included in Definition 4.1 and Definition 4.2 to take care of election situations involving ties. These definitions tell us that no matter how ties are broken prior to the merger/division, it is possible to break them in such a way after the merger/division that the actual constituency/constituencies do not lose. To handle a merger or division which involves more than two constituencies, we use the actual definition iteratively.

Theorem 4.1
HA is the unique divisor method which encourages merger.

Theorem 4.2
SD is the unique divisor method which encourages division.

These theorems are stated by [B&Y] (page 150-151). [B&Y] (page 150) prove Theorem 4.1 by using the condition from Definition 4.1 iteratively. Below we present a proof of Theorem 4.2 along the same lines as their proof of Theorem 4.1:

Proof: We start by demonstrating that SD does encourage division: To obtain a SD apportionment after a division, we begin by using the same value for the common divisor z_A as before the division. The sum of the two newly formed constituency quotients $\frac{p_i}{z_A} + \frac{p_s}{z_A}$ is equal to the constituency quotient $\frac{p_i + p_s}{z_A}$ before

the division. In the following we assume that ties are broken the same way after the division as they were before the division. This assumption does not violate the premise in Definition 4.2. SD rounds constituency quotients up to the nearest integer to determine the current assignment. Since $\lceil \frac{p_i}{z_A} \rceil + \lceil \frac{p_s}{z_A} \rceil \geq \lceil \frac{p_i + p_s}{z_A} \rceil$, i and s are never awarded fewer seats than before the division, and they are sometimes awarded one seat more. An extra seat to i or s means that there has been awarded one seat too much for the whole election situation. Then z_A must be increased until some constituency loses a seat, i.e. until some constituency quotient falls below a divisor $d_l = l - 1$. i or s may be the constituency which loses this seat. In any case i and s have not lost compared with the election situation before the division.

Next we prove that SD is the unique divisor method which encourages division: Let A be a strict divisor method and suppose that it encourages division. Then consider the following election situation $(\hat{\mathbf{p}},h)$ with two constituencies: The population vector is $\hat{\mathbf{p}} = [\zeta, \zeta]$, where ζ is a positive real number. There are a total of $h = 2 \cdot b \cdot (m - 1) + 1$ seats for distribution, where $b \geq 1$ and $m \geq 2$ are integers. Since a strict divisor method is balanced, there exists an apportionment $\hat{\mathbf{a}} = [b \cdot (m - 1), b \cdot (m - 1) + 1]$ for the election situation $(\hat{\mathbf{p}},h)$. We form $m - 1$ new constituencies by dividing the second constituency in $m - 1$ identical parts with the population vector $\tilde{\mathbf{p}} = [\zeta, \frac{\zeta}{m-1}, \ldots, \frac{\zeta}{m-1}]$ as the result. A encourages division so there exists an apportionment for which $\sum_{i=2}^{m} a_i \geq b \cdot (m - 1) + 1$ and $a_1 \leq b \cdot (m - 1)$. By exactness, $\mathbf{a} = [b \cdot (m - 1), b, \ldots, b]$ is the unique apportionment for the smaller house size $\bar{h} = 2 \cdot b \cdot (m - 1)$. Since A is house monotone, $a_1 \geq b \cdot (m - 1)$ for the house size $h = 2 \cdot b \cdot (m - 1) + 1$. A is balanced so one of the constituencies with population $\frac{\zeta}{m-1}$ gets $(b + 1)$ seats while the remaining get b seats each. Therefore, $\mathbf{a} = [b \cdot (m - 1), b, \ldots, b, (b + 1)]$ is an apportionment for the election situation $(\tilde{\mathbf{p}},h)$. From the apportionment to the first and last constituency we get the inequality:

$$\frac{\zeta}{d_{[b \cdot (m-1) + 1]}} \leq \frac{\frac{\zeta}{m-1}}{d_{(b+1)}} \tag{4.1}$$

which by making use of the interval condition from equation (2.9) implies:

(4.2) $$d_{(b+1)} \leq \frac{d_{[b\cdot(m-1)+1]}}{m-1} \leq \frac{b\cdot(m-1)+1}{m-1} = b + \frac{1}{m-1}$$

When the number of constituencies $m \rightarrow \infty$, $d_{(b+1)} \rightarrow b$, which proves that only SD encourages division. Theorem 3.3, Theorem 3.4, Theorem 4.1, and Theorem 4.2 mean that HA and SD can be seen as the extreme opposites within the class of strict divisor methods.

[B&Y] (page 152) define **coalition-neutrality** the following way: A divisor method is coalition-neutral if a coalition of two constituencies is just as likely to gain a seat as to lose one. Let us extend coalition-neutrality to all apportionment methods and define it in terms of probabilities. Below Pr[K] denotes the probability of an event K.

Definition 4.3
A is **coalition-neutral** if for an arbitrary election situation $(\mathbf{p},h)$
$\Pr[\hat{a}_{(i+s)} = a_i + a_s + 1] = \Pr[\hat{a}_{(i+s)} = a_i + a_s - 1]$ for all $i \neq s$.

Let $\hat{r}_i$ denote the remainder of a constituency quotient: $\hat{r}_i = \frac{p_i}{z} - d_{a_i}$ where a_i is the final apportionment to constituency i. For all constant parametric divisor methods $\hat{r}_i \in [0, 1)$, and it seems reasonable to assume that the remainders are independently and uniformly distributed between 0 and 1. Given a similar assumption, [B&Y] (page 152-153) show that MF is approximately coalition-neutral. Moreover, they state the proposition that MF is the unique coalition-neutral divisor method.

Let E[k] denote the expected value of a variable k. The fractional parts of the quotas $r_i \in [0, 1)$, so a natural assumption is that these fractions are independently and uniformly distributed between 0 and 1. This assumption implies that the expected value of an arbitrary fraction is $E[r_i] = 0{,}5$. Furthermore, the expected value of the smallest fraction which qualifies for a seat with LF, r^*, is also equal to 0,5, $E[r^*] = 0{,}5$. By utilizing the same arguments as [B&Y] (page 153) use for MF, it can be shown that LF is approximately coalition-neutral.

Definition 4.3 says something about situations where the difference between $\hat{a}_{(i+s)}$ and $a_i + a_s$ is one seat. With strict divisor methods the difference between these two apportionment figures is at most one seat if ties are broken the same way in both election situations. **Stability**, [B&Y] (page 151), is a condition which restricts the difference between $\hat{a}_{(i+s)}$ and $a_i + a_s$ to at most one seat:

Definition 4.4
A is **stable** if $\mathbf{a} \in A(\mathbf{p},h)$ implies the existence of $\hat{\mathbf{a}} \in A(\hat{\mathbf{p}},h)$
where $a_i + a_s + 1 \geq \hat{a}_{(i+s)} \geq a_i + a_s - 1$ for all $i \neq s$ and $(\mathbf{p},h)$.

In words: An apportionment method is stable if a coalition of two constituencies does not win or lose more than one seat compared with their combined total prior to the merger. All apportionment methods considered so far are stable.

4.2 The distribution of remainders and fractions

To reach the results following Definition 4.3 it was assumed that the remainders of the constituency quotients $\hat{r}_i$ and the fractional parts of the quotas r_i were uniformly distributed between 0 and 1. In this section we test these assumptions empirically by utilizing the election situations for the 13 countries in Table 2.3. We apply the chi-square test with a 5% level of significance. The two tests below may be viewed as tests of "the obvious". Our reason for including them is that we have not seen the assumptions being tested before.

The two tests are quite similar, only the way we prepare the data sets differ. Let us explain the test for the fractions first: Our null hypothesis is that r_i is uniformly distributed over the interval $[0, 1\rangle$. The alternative hypothesis is that the distribution is not uniform. We divide the interval $[0, 1\rangle$ in n subintervals of even length such that the expected number of fractions in each subinterval is at least 5. Then we can use approximation to the chi-square distribution with n - 1 degrees of freedom, [Li] (page 220-222). For Austria and Iceland, which each has

less than 11 constituencies, we construct a composite set. The fractions from all countries could have been collected in one data set, but we have chosen not to do so since Japan with its large number of constituencies would have been too influental. Since the fractions for an election situation sum to an integer, the last fraction is determined by the m - 1 other fractions. We make sure that the fractions we utilize in the test are independent by randomly eliminating one fraction from each election situation. Our test statistic is:

$$Q = \sum_{a=1}^{n} \frac{(X_a - \frac{m-1}{n})^2}{\frac{m-1}{n}} \tag{4.3}$$

where X_a is the observed number of fractions in subinterval a. Q is chi-square distributed with n - 1 degrees of freedom if the null hypothesis is correct.

We move on to the test for the remainders. Our null hypothesis for every constant parametric divisor method is that $\hat{r}_i$ is uniformly distributed over the interval [0, 1⟩. We test this hypothesis for SD, MF, and HA using the same test as for the fractions, but with a minor modification regarding the preparation of data sets: From equation (2.11) we know that the common divisor z can be chosen within an interval. For each election situation we choose the highest possible common divisor which distributes the right number of seats, i.e. $z = q_h$. The remainder for the constituency which wins the last seat in parliament is always 0 with this procedure. This violates the assumption of independence so we eliminate this remainder from each data set.

We have calculated the test statistic Q for each combination of data set and apportionment method. Based on the degrees of freedom for the actual data set, the P-value for the observed Q is determined. These P-values are presented in the right part of Table 4.1. Furthermore, the table contains two columns with headings EX and DOF respectively. We have calculated the figures in these columns as follows: The expected number of fractions/remainders in each subinterval is EX = $\frac{m-1}{n}$ ($\frac{m-2}{n}$ for the set consisting of Austria and Iceland), while the number of degrees of freedom is DOF = n - 1.

Table 4.1

Country	Year	Based on	Seats	*m*	n	EX	DOF	LF	SD	MF	HA
Japan	1985	Population	512	130	20	6,450	19	0,98	0,14	0,51	0,43
Thailand	1970	Population	219	71	10	7,000	9	0,11	0,08	0,09	0,82
Greece	1981	Population	300	56	8	6,875	7	0,62	0,76	0,97	0,14
USA	1990	Population	435	50	8	6,125	7	0,50	0,40	0,77	0,03
Sweden	1990	Voters	349	28	4	6,750	3	0,12	0,43	0,12	0,87
Switzerland	1990	Population	200	26	4	6,250	3	0,23	0,03	0,39	0,27
Norway	1989	Voters	165	19	3	6,000	2	0,51	0,61	0,85	0,51
Denmark	1990	Voters	175	17	2	8,000	1	0,32	1,00	0,62	1,00
Germany	1990	Voters	656	16	2	7,500	1	0,44	0,80	0,80	0,44
ICE + AUT	1987/91	Vot./Pop.	63/183	8 + 9	2	7,500	1	0,80	0,20	0,80	0,80
Finland	1991	Voters	200	15	2	7,000	1	1,00	0,59	1,00	0,29
Canada	1981	Population	282	12	2	5,500	1	0,37	0,76	0,76	0,37

EX = Expected number in each subinterval DOF = Degrees of freedom

P-values for the test of uniform distribution of fractions (LF) and remainders (SD, MF, and HA) between 0 and 1.

The null hypothesis for the fractions is not rejected for any of the 12 data sets. However, the null hypothesis for the remainders is rejected for the US with HA and for Switzerland with SD, i.e. for 2 out of 36 cases. A rejection percentage for the remainders test of just above 5% is about what to expect with a level of significance equal to 5%. The tests above are based on a relatively small sample, but our conclusion is clear: Both fractions and remainders are uniformly distributed between 0 and 1.

4.3 Treatment of small versus large constituencies

In the following we focus on pairs of constituencies for which $p_i < p_s$. It is reasonable to say that an apportionment method **favours small constituencies compared to** another apportionment method if the two methods give the same number of seats to this subset of two constituencies and the first method gives the smallest constituency in the subset at least as many seats as the other method does, [H] (page 83):

Definition 4.5
A **favours small constituencies compared to** A^*
if $\mathbf{a} \in A(\mathbf{p},h)$, $\mathbf{a}^* \in A^*(\mathbf{p},h)$, $p_i < p_s$, and $a_i + a_s = a_i^* + a_s^*$
imply $a_i \geq a_i^*$ for all $(\mathbf{p},h)$.

Observe that the definition says nothing about situations where the two apportionment methods give different seat totals to the subset consisting of i and s, i.e. situations where $a_i + a_s \neq a_i^* + a_s^*$. Moreover, because we usually assume that ties are broken arbitrarily, A will not generally favour small constituencies compared to itself. Definition 4.5 is symmetric so if A favours small constituencies compared to A^*, then A^* favours large constituencies compared to A. We can rank some of the divisor methods in the matter of favouritism:

Theorem 4.3
The divisor method A favours small constituencies compared to
the divisor method A^* if $\frac{d_l}{d_s} > \frac{d_l^*}{d_s^*}$ for all integers $l > s \geq 1$
and where $d_1 > 0$ and $d_1^* > 0$.

For proof of Theorem 4.3 see [H-A] (page 84) or [B&Y] (page 118). From the divisor ratio $\frac{d_l}{d_s}$ we can see how fast the divisors grow. The faster they grow the more favourable is the divisor method for small constituencies.

If we define $\frac{d_l}{0} > \frac{d_l^*}{d_1^*}$ if $d_1^* > 0$ and $\frac{d_l}{0} > \frac{d_l^*}{0}$ if $d_l^* > d_l > 0$, Theorem 4.3 also holds for divisor methods with $d_1 = 0$ and $d_1^* = 0$. The last part of this additional definition contradicts what one might expect, but is necessary to rank for example HM and EP. It is clear that HM favours small constituencies compared to EP for divisor ratios where $s \geq 2$. However, if we had defined the relationship between divisor ratios involving $d_1 = 0$ another way, the two methods could not have been ranked by the premise in Theorem 4.3. With the additional definitions regarding instances with $d_1 = 0$ we have the following transitive ranking of the five traditional divisor methods: SD > HM > EP > MF > HA, where the method to the left of the > sign is more favourable to small constituencies than the method to the right.

We cannot rank all apportionment methods, not even within the class of strict divisor methods. The reason is that for some election situations a method may be favourable for small constituencies relative to the other method, while for some other election situations the opposite might be true. DM has not been included in the ranking above because it cannot be ranked versus HM and EP: The first divisor of HM and EP, $d_1 = 0$, is more favourable to small constituencies than DM's first divisor, $d_1^* = \frac{1}{3}$, because $\frac{d_l}{0} > \frac{d_l^*}{\frac{1}{3}}$ for every $l \geq 2$. However, for other divisor ratios the favouritism goes the other way. By restricting our attention to divisor ratios where s and l are consecutive integers, we find that the ratio for DM is larger than EP's ratio when $s = 2$ and $l = 3$, while it is equal to HM's ratio when $(s, l) = (3, 4)$ and larger when $(s, l) = (4, 5)$.

Within the family of constant parametric divisor methods it is easy to rank the methods. Manipulation of a divisor ratio for such a method yields:

$$\frac{d_l}{d_s} = \frac{l - 1 + t}{s - 1 + t} = \frac{l - s + (s - 1 + t)}{s - 1 + t} = 1 + \frac{l - s}{s - 1 + t} \tag{4.4}$$

The special case with $s = 1$ and $t = 0$, i.e. zero in the denominator, is covered by the definition following Theorem 4.3. To study the impact of the parameter t on the divisor ratio we find the derivative with respect to t:

(4.5) $$\frac{\partial \frac{d_l}{d_s}}{\partial t} = \frac{s - l}{(s - 1 + t)^2}$$

The numerator in (4.5) is negative, while the denominator is positive with one exception; for the combination $s = 1$ and $t = 0$ it is zero. Thus, every divisor ratio $\frac{d_l}{d_s}$ is a strictly decreasing function of $t \in [0, 1]$. This means that the constant parametric divisor methods can be ranked by simply looking at t:

Theorem 4.4
CP_t favours small parties compared to CP_t^* if and only if $t < t^*$.

This result is also given in [H-A] (page 85). It means that the transitive ranking of the CP_t divisor methods we have encountered is: SD > DM > MF > HA. It is interesting to notice that SD which encourages division also favours small constituencies compared to all other constant parametric divisor methods, while HA which encourages merger also favours large constituencies compared to all other constant parametric divisor methods.

After these rankings of divisor methods it is interesting to look at the relationship between LF and divisor methods in the matter of favouritism:

Theorem 4.5
SD is the unique constant parametric divisor method which favours small constituencies compared to LF.

Theorem 4.6
HA is the unique constant parametric divisor method which favours large constituencies compared to LF.

Below we prove Theorem 4.5. We start by proving that SD favours small constituencies compared to LF and follows up by proving that SD is the only constant parametric divisor method which does so. The proof of Theorem 4.6 is similar and is not shown.

Proof: We study an arbitrary election situation where $m \geq 2$. Our focus is on the two constituencies i and s, where the population of the latter is larger, i.e. $p_s > p_i$ which further implies $q_s > q_i$. To prove that SD favours small constituencies compared to LF, we compare their apportionments to i and s:

We start with the LF apportionment for the whole election situation, i.e. for all m constituencies. Our two constituencies i and s are at least apportioned $\lfloor q_i \rfloor$ and $\lfloor q_s \rfloor$ seats respectively. The fractional parts of their quotas, r_i and r_s, will qualify for a total of zero, one, or two seats dependent on the threshold r^* for this particular election situation.

We now switch our attention to SD. In the following we utilize terminology and results from section 2.7. As the initial value for z_A, the common divisor for SD, we use the national average population per seat p. This means that constituency quotients and quotas are identical at the outset, i.e. $\frac{p_i}{z_A} = q_i$ and $\frac{p_s}{z_A} = q_s$. All constituency quotients are rounded upwards to the nearest integer to get the SD apportionment, i.e. i and s are assigned $\lceil q_i \rceil$ and $\lceil q_s \rceil$ seats initially. The result is that too many seats are distributed when $z_A = p$; the only exception is the case of exact proportionality where the right number of seats is distributed. By increasing z_A from p, all constituency quotients are decreased. To determine the SD apportionment for the whole election situation, z_A is increased until exactly h seats are distributed. Below we explain what happens to the remainders of the constituency quotients during this adjustment process:

The remainder for constituency i with SD is $\hat{r}_i = \frac{p_i}{z_A} - d_{a_i} = \frac{p_i}{z_A} - a_i + 1$, where a_i is the current assignment to i. At some stage of the adjustment process some constituency quotient becomes equal to one of the divisors $d_l = l - 1$, which means that the remainder of this constituency quotient is equal to zero. When z_A is increased just a little bit more, the constituency quotient becomes smaller than the divisor d_l, and the constituency loses a seat. We say that the remainder vanishes since the constituency quotient gets a kind of new remainder. How many remainders which have to vanish before an appropriate z_A value is reached depend on the actual election situation. Our primary concern is what happens to

the remainders for constituencies i and s. We investigate this matter below.

When z_A is increased, all constituency quotients decrease by the same relative amount. This means that the remainder for the largest of our constituencies $\hat{r}_s$ decreases more in absolute terms than the remainder for our smallest constituency $\hat{r}_i$. At the starting point, with $z_A = p$, the remainders are equal to the fractional parts of the quotas, i.e. $\hat{r}_i = r_i$ and $\hat{r}_s = r_s$. There are two possibilities regarding the initial relationship between the remainders (fractions) for our two constituencies, namely $r_i \geq r_s$ and $r_i < r_s$. Below we investigate in which succession the remainders for i and s vanish given that the appropriate adjustment of z_A makes them vanish. $\hat{r}_s$ decreases faster than $\hat{r}_i$ so when $\hat{r}_i$ is not smaller than $\hat{r}_s$ at the start, i.e. $r_i \geq r_s$, $\hat{r}_s$ vanishes first. Moreover, $\hat{r}_s$ may vanish twice before $\hat{r}_i$ vanishes for the first time, but this depends on the relative size difference between the quotas (initial constituency quotients) q_i and q_s. Even with a smaller $\hat{r}_i$ than $\hat{r}_s$ initially, i.e. $r_i < r_s$, $\hat{r}_s$ may vanish twice before $\hat{r}_i$ vanishes for the first time. When $r_i < r_s$ it is possible that $\hat{r}_i$ vanishes first. Since $\hat{r}_s$ decreases more in absolute terms than $\hat{r}_i$ does, it is impossible for $\hat{r}_i$ to vanish twice before $\hat{r}_s$ vanishes for the first time. Let us combine the information above with the LF apportionment to the fractions r_i and r_s. Table 4.2 summarizes the possible outcomes for different combinations:

Table 4.2

Total number of seats to r_i and r_s with LF	$r_i \geq r_s$	$r_i < r_s$
2 seats	-No change -Fewer seat(s) to $i \cup s$	-No change -Fewer seat(s) to $i \cup s$
1 seat	-No change -A different number of seats to $i \cup s$	-No change -i gets one more seat and s one less -A different number of seats to $i \cup s$
0 seats	-No change -i gets one more seat and s one less -A different number of seats to $i \cup s$	-No change -i gets one more seat and s one less -A different number of seats to $i \cup s$

Change in the apportionment to the subset consisting of constituencies i and s when we go from LF to SD.

The possible outcomes can be classified in three groups:

1) The same apportionment to i and s with LF and SD.
2) A different number of seats to $i \cup s$ with SD than with LF.
3) SD gives i one seat more and s one seat less than LF does.

Since we only are interested in election situations where LF and SD give the same number of seats to the subset consisting of constituencies i and s, consult Definition 4.5, group 2) is not of interest here. Below follows a brief description of how the two other groups occur:

If a remainder vanishes once if the corresponding fraction is not awarded a seat or does not vanish if the corresponding fraction is awarded a seat, the actual constituency wins the same number of seats with SD as with LF. When this happens to both i and s, we get group 1). Group 3) occurs when the remainder $\hat{r}_s$ vanishes twice, $\hat{r}_i$ does not vanish, and SD gives the same number of seats to $i \cup s$ as LF does.

All election situations where SD and LF give the same number of seats to the subset consisting of i and s have now been covered. For no situation will SD give less seats to i than LF does, and it may give i more seats. This proves that SD favours small constituencies compared to LF.

Above we demonstrated what happens when the common divisor z is increased from its initial value of p. When HA is the divisor method, z_J has to be decreased from its intial value of p. The only exception to this rule is the case of exact proportionality, where no change in z_J is needed. In fact, in the case of exact proportionality $z = p$ will do for all exact divisor methods, i.e. all methods which satisfy Definition 2.2. Moreover, their apportionment will be equal to the LF apportionment. The effect of lowering z is the opposite of the effect of heightening z. That HA favours large constituencies compared to LF is proved this way.

Notice that for both SD and HA the direction of the needed change in z from its initial value of $z = p$ is independent of the election situation, as explained in section 2.7. For all other constant parametric divisor methods the direction of the needed change in z depends on the actual election situation. To prove the uniqueness of SD we construct an election situation where all other constant parametric divisor methods fail to favour small constituencies compared to LF:

We face an election situation with $m \geq 2$ where the quota sizes of the constituencies are as follows: Constituency i has a quota of $q_i = l - 1 + \frac{1}{m}$, the larger constituency s has a quota of $q_s = L - 1 + \frac{1}{m}$ where $L > l$, while the quota of each of the other constituencies is $l - 1 + \frac{1}{m}$. Thus, any constituency may win the only seat distributed to the fractions with LF. Let us assume that constituency i is awarded this seat. Then consider a constant parametric divisor method with $t > 0$ which we suppose favours small constituencies compared to LF. When $t > \frac{1}{m}$, i.e. $d_l > l - 1 + \frac{1}{m}$, only $h - 1$ seats are distributed with $z = p$. Thus, z has to be decreased, with the result that constituency s wins its L-th seat. Even the possibility of s winning its L-th seat is a violation of the premise in Definition 4.5. Hence, for a constant parametric divisor method to favour small constituencies compared to LF its parameter must satisfy the following condition:

$$t < \frac{1}{m} \tag{4.6}$$

When $m \rightarrow \infty$ the term on the right hand side approaches 0, which means that only $t = 0$ (SD) guarantees that constituency s is not awarded its L-th seat. This contradicts the earlier assumption that a constant parametric divisor method with $t > 0$ favours small constituencies compared to LF, thereby proving that SD is the unique constant parametric divisor method which favours small constituencies compared to LF. The uniqueness of HA among the constant parametric divisor methods regarding favouring of large constituencies compared to LF is proved by utilizing an election situation where the fractional parts of all quotas are $\frac{m-1}{m}$.

The reason why the two theorems above only concern the relationship between the family of constant parametric divisor methods and LF is that there exist "house size dependent" divisor methods which favour small (large) constituencies compared to LF. Below we present two examples of such methods:

Consider the divisor series: $d_l = l - 1 + \frac{l-1}{h-1}$. This series defines a divisor method which can be described as an "even step stairway parametric divisor method". It can be proved that this method favours small constituencies compared to LF. However, multiplication of the divisor series by the factor $\frac{h-1}{h}$ reveals that the method is SD in disguise!

To find a "house size dependent" divisor method which really is different from SD and favours small constituencies compared to LF, we look at a divisor series which is equal to the series for SD with the exception of one divisor d_a. This divisor is bounded the following way: $a - 1 < d_a < a$, where the upper bound follows from Definition 2.2 and the fact that $d_{(a+1)} = a$. We let $h \geq a > \frac{h}{2}$ such that only the largest constituency may be affected by d_a directly. However, that it is more difficult for the largest constituency to win its ath seat is to the smaller constituencies' advantage. Example 4.1 shows possible consequences of the use of such a divisor method:

Example 4.1

We face an election situation where three constituencies, labelled A, B, and C, compete for 9 seats. These constituencies have quotas of $q_A = 4{,}9$, $q_B = 3{,}0$, and $q_C = 1{,}1$ respectively. Consider the divisor method defined by the same divisor series as SD, but with the exception that $d_5 = 4{,}95$ instead of 4. The apportionment with this method plus the apportionment with LF and SD are given in Table 4.3 below:

Table 4.3

Constituency	Quota	LF	SD	$d_5 = 4{,}95$
A	4,9	5	5	4
B	3,0	3	3	3
C	1,1	1	1	2

Apportionments with LF, SD, and the divisor method defined by the divisor series:
$d_1 = 0, ..., d_4 = 3, d_5 = 4{,}95, d_6 = 5, ..., d_9 = 8$.

The LF apportionment is easily found from the quotas. The SD apportionment is identical with the LF apportionment here because $\frac{q_A}{d_5} > \frac{q_C}{d_2} > \frac{q_B}{d_4}$. There is a different story with the other divisor method, because then $\frac{q_C}{d_2} > \frac{q_B}{d_4} > \frac{q_A}{d_5}$. This demonstrates that there exist divisor methods which are more favourable to small constituencies than SD for some election situations. As mentioned in connection with Theorem 4.3, the premise for a divisor method to favour small constituencies compared to another divisor method is only satisfied for a subset of the divisor methods.

It is debatable whether a "house size dependent" divisor method can be characterized as a complete divisor method because its divisor series changes with the house size. Consider the following divisor series for a house size of 10: $d_l = l - 1$ for $l \in \{1, ..., 5, 7, ..., 10\}$ and $d_6 = 5{,}95$. Have this divisor series enough in common with the divisor series in Example 4.1 to be characterized as the "same" method? With respect to Definition 4.5 the answer has to be no, which clears the ground for the following propositions:

Proposition 4.1
SD is the unique divisor method which favours small constituencies compared to LF.

Proposition 4.2
HA is the unique divisor method which favours large constituencies compared to LF.

Chapter 5: Bias

The topic of this chapter is bias of vector apportionment methods. To measure the bias we need to group the constituencies. We call methods used for this task division methods. In the first section we introduce the bias problem and present some basic terminology regarding division methods. During the chapter we present four division methods, namely number division in section 5.2, quota division in section 5.3, size division in section 5.5, and cluster division in section 5.6. To get an impression of the difficulties these methods have to deal with we take a look at the distribution of constituency populations in section 5.4. Section 5.7 illustrates the application of the four division methods on a real election situation. In the last section we carry out an empirical test of bias.

5.1 Introduction and basic grouping terminology

An apportionment method which regularly favours any group of constituencies is biased. For an apportionment method to be unbiased, it must give all groups of constituencies their exact quota in the long run. By the apportionment to a group we mean the sum of the seats given to the individual constituencies in this group.

The debate regarding how to measure bias of vector apportionment methods has based itself on the US experience, see [B&Y] (page 118→) and [E]. In their empirical testing [B&Y] utilize US data from 19 censuses. To give a broader foundation for conclusions about bias, we utilize data from 13 different countries, though only one data set from each, in our empirical test in section 5.8.

How do we pick out constituencies which "belong" to a specific group such that it is reasonable to say that this group comprises the same kind of constituencies for different election situations? Our answer to this question is methods which group constituencies in a consistent manner. Since the process of grouping constituencies can be seen as a process where the set of all constituencies M is divided in a predetermined number of groups, we call methods used for the grouping process **division methods**.

Our starting point for all division methods is the constituencies sorted in descending order based on population. The constituencies shall be divided in c disjoint groups. We use the index g for the groups and denote the set of all groups C. Hence, $g \in C = \{1, ..., c\}$. The number of constituencies in group g is denoted n_g. We gather all such numbers in the grouping vector $\mathbf{n} = (n_1, ..., n_g, ..., n_c)$. Obviously, $\sum_{g \in C} n_g = m$. Each group must have at least one member, which means that $n_g \geq 1$ and $c \leq h$. In the bias test in section 5.8 we divide the constituencies in 2, 3, and 4 groups, i.e. $c \in \{2, 3, 4\}$. We let G stand for the set of constituencies which belong to group g. The ordering of the constituencies means that $g = 1$ is the group of the largest constituencies while $g = c$ is the group of the smallest constituencies. Another consequence of the ordering is that the smallest constituency in one group is never smaller than the largest constituency in the next. To find out which group a constituency belongs to, one needs to know its placing on the sorted constituency list. Constituency nos. 1 - n_1 belong to the first group, nos. $(n_1 + 1) - (n_1 + n_2)$ to the second group, ..., and nos. $(m - n_c + 1) - m$ to the cth (last) group.

[B&Y] (page 126) eliminate states (constituencies) with a quota of less than 0,5 from their test. We have not eliminated such constituencies. One reason for this choice is that we ignore law determined minimum requirements in our test. However, small constituencies may lead to violation of guideline 2) below. How such violations happen is explained later. What requirements should a division method used for bias analysis satisfy? We suggest the following rather weak guidelines:

1) Because population size is the basis for the grouping, the members of a group should be as equal as possible sizewise.
2) The total quota of each group should be of some magnitude in order to avoid situations where a group is not assigned any seat.

5.2 Number division

[B&Y] (page 126) divide the constituencies evenly in three disjoint groups: Large, medium, and small, where the middle group takes up the extras if the number of constituencies m is not divisible by three. The constituencies may also be divided in another number of disjoint groups. Our division in the same spirit as [B&Y] is carried out in a slightly different way since we distribute the extras one at a time, starting with the group of the smallest constituencies. We call this way of creating groups **number division**, because its objective is to make the number of constituencies in each group as equal as possible. Mathematically it can be described as:

Algorithm 5.1

Step 1: Determine the maximal whole number of constituencies which can be assigned to each group $\overline{m}$ and the remainder of this division $\bar{r}$:

(5.1)
$$\overline{m} = m \text{ div } c$$
$$\bar{r} = m \bmod c$$

Step 2: For $g = 1, \ldots, c$ let the group consist of the following number of constituencies:

(5.2)
$$n_g = \overline{m} \quad \text{if } g \leq c - \bar{r}$$
$$n_g = \overline{m} + 1 \quad \text{if } g > c - \bar{r}$$

Thus, the number of constituencies in a group deviates by at most one from the number in any other group. A consequence is that the group of the largest constituencies usually contains much more seats than the group of the smallest constituencies. Number division does not generally satisfy any of the division guidelines. The advantage of number division is its simplicity.

5.3 Quota division

The objective of **quota division** is to divide the constituencies such that the quota sizes of the groups are as equal as possible. Let $\bar{q} = \frac{h}{c}$ denote the average quota per group and $q(g) = \sum_{i \in G} q_i$ the total quota of the constituencies assigned to group g.

A possible objective function for quota division is found by squaring the deviations between group quotas and the average group quota and minimizing the sum of these squared deviations, i.e. $\min_{\mathbf{n}} \sum_{g \in C} [q(g) - \bar{q}]^2$. By eliminating constant terms it can be shown that this objective function is equivalent to:

$$\min_{\mathbf{n}} \sum_{g \in C} [q(g)]^2 \tag{5.3}$$

However, we have chosen another objective function. We minimize the maximal deviation between the quota of a group and the average group quota:

$$\min_{\mathbf{n}} \max_{g \in C} |q(g) - \bar{q}| \tag{5.4}$$

We use (5.4) lexicographically. This means that after the group with the minimum maximal deviation has been located, (5.4) is used again, if necessary, to locate the next group etc. Because the groups are interdependent, the use of (5.4) once is often enough to determine the whole grouping vector. Quota division is trivial when c is equal to 1, 2, m-1, or m. We determined the quota divisions utilized in the empirical test in section 5.8 by trying and failing. This was

manageable since the highest number of groups was as small as 4. We have later developed a Pascal program for quota division, which can be found as a part of the MatrixBias program in Appendix 2.

Quota division can be formulated as a simple version of the political districting problem, see [G&N]. In addition to constraints (6) - (8) in [G&N] (page B-497), we include the condition:

$$\min_{i \in G} q_i \geq \max_{s \in (G+1)} q_s \qquad \text{where } g \in \{1, ..., c - 1\} \tag{5.5}$$

Thus, the smallest constituency in one group should never be smaller than the largest constituency in the next. This is a kind of contiguity condition in the terminology of the political districting problem. [G&N] solve the problem of determining political districts which minimize the maximal deviation in two phases. In the first phase they generate feasible districts, before they in the second phase optimize. Below we explain how we generate feasible groups:

For use in Algorithm 5.2 below we introduce the following notation and terminology: With the constituencies sorted in descending order we let (x,y), where x, y $\in M$ and $1 \leq x \leq y \leq m$, denote the group consisting of all constituencies from x to and including y. $q_{(x,y)}$ denotes the total quota of this group. The absolute value of the deviation from the average group quota $|q_{(x,y)} - \bar{q}| = |q(g) - \bar{q}|$ plays an important role in quota division. The determination of a tight upper bound for such deviations speeds up the solution process. Let ϖ denote such an upper bound. We define a group (X,Y) as **connected to a preceding group** (x,y) if the largest constituency in (X,Y) is the immediate neighbour of the smallest constituency in (x,y), i.e. if X = y + 1. Two connected groups are clearly disjoint. Together they contain all constituencies from x to and including Y. We define a string of c connected groups which together contain all m constituencies as a **connected tree of groups**.

Algorithm 5.2 below consists of three steps. In the first step we determine a weak upper bound ϖ and generate candidates for the first group ($g = 1$). Candidates for the other groups are generated in step 2. In the last step we eliminate candidates which are not part of a connected tree of groups.

Algorithm 5.2

Step 1: Starting from the top of the sorted constituency list we determine the highest number of constituencies z for which the group quota is lower than the average group quota, i.e. z is maximized subject to $q_{(1,z)} - \bar{q} < 0$. From the determination of z it follows that $q_{(1,z+1)} - \bar{q} \geq 0$. By expressing the group quota $q_{(1,z+1)}$ as $q_{(1,z)} + q_{(z+1)}$ we get:

$$q_{(z+1)} \geq \bar{q} - q_{(1,z)} \tag{5.6}$$

Because the constituency sizes are non-increasing, it is impossible for a group further down the list to be part of an optimal solution if its size deviates from the average group quota by more than $q_{(z+1)}$. We therefore set the (initial) upper bound equal to the size of constituency $z + 1$, i.e. $\varpi = q_{(z+1)}$. Next we generate candidates for $g = 1$ (the first group). All groups (1,y) for which constituency y is chosen such that $|q_{(1,y)} - \bar{q}| \leq \varpi$ are candidates for $g = 1$.

Step 2: For every candidate (x,y) for group $g - 1$ we generate candidates for group g as follows: A candidate (X,Y) for g must be connected to the candidate for $g - 1$, so the first constituency is $X = y + 1$. All groups (X,Y) for which constituency $Y \leq m$ is chosen such that $|q_{(X,Y)} - \bar{q}| \leq \varpi$ are candidates for g. The upper bound ϖ may be strengthened by considering deviations regarding candidates for $g = 2$ etc, but we have not elaborated this possibility. We repeat step 2 until candidates for group $g = c$ (the last group) have been generated.

Step 3: The candidates for the different groups have been generated by starting from the top of the sorted constituency list. We must check whether they are feasible starting from the bottom of the list. A candidate for the last group must include the smallest constituency, i.e. $Y = m$, to be feasible. Every candidate

which does not "reach the bottom" of the constituency list is eliminated. Furthermore, a candidate for group g - 1 must be connected to a feasible candidate for group g to be feasible itself. We eliminate all infeasible candidates. To summarize: All candidates which are not part of a connected tree of groups are infeasible and are therefore eliminated.

Algorithm 5.2 is exemplified in section 5.7. The candidates may be represented by a matrix. Each row of this matrix represents a constituency, while each column represents a candidate. The entries, which we denote h_{is}, are either 0 or 1. h_{is} is equal to 1 if candidate s includes constituency i and equal to 0 if not. Divisions with minimum maximal deviation can be found with the algorithm presented in [G&N] (page B-502). Moreover, this algorithm can be used to determine all remaining minimum maximal deviations.

We end this section with some comments regarding optimal quota divisions: Because group members become smaller and smaller as we move down the sorted constituency list, it is usually easier to bring a group of small constituencies close to the average group quota than a group of large constituencies. This explains why the maximal deviation often occurs for $g = 1$. Given that all constituency sizes are different, the number of constituencies in a group will never decrease with increasing group number, i.e. $n_g \leq n_{(g+1)}$ where $g \in \{1, ..., c - 1\}$. This follows from the initial ordering of the constituencies and the objective of quota division. The strong point of quota division is that all groups contain about the same number of seats, which means that division guideline 2) is satisfied. On the negative side; quota division does not necessarily satisfy division guideline 1). Moreover, it requires more work than number division.

5.4 The population distribution

Our empirical bias test in section 5.8 utilizes data from 13 countries. Different countries mean different distributions of constituency populations. Figure 5.1 and Figure 5.2 below show two of the most extreme distributions in our sample. The measures used for the axes in these figures need an explanation:
We determine the normalized (population) size of a constituency by dividing its population by the population of the most populous constituency, i.e. the constituency at the top of the sorted constituency list:

(5.7) $$\text{Normalized size of constituency } s = \frac{p_s}{\max_{i \in M} p_i}$$

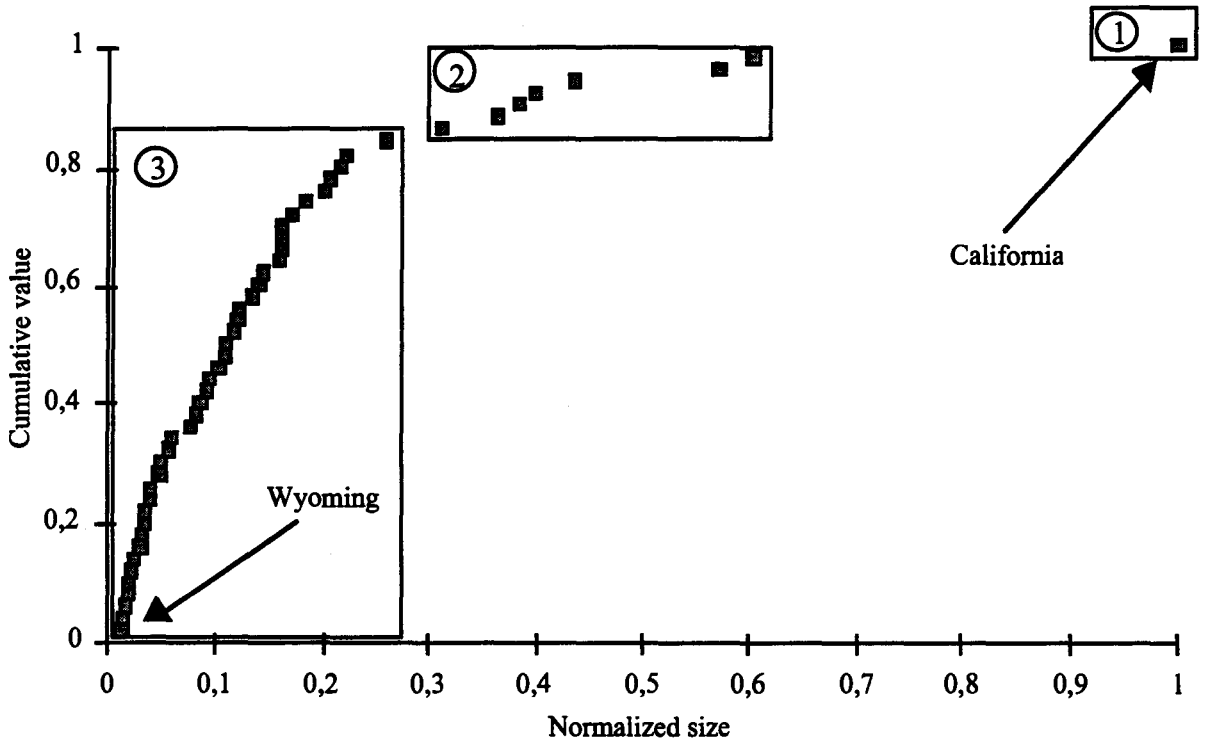
Thus, the largest constituency has a normalized size of 1, and the normalized size of any constituency is within the interval ⟨0, 1]. The normalized sizes could have been illustrated in a one-dimensional diagram. We find it more informative to illustrate them as a kind of cumulative distribution function. The cumulative values, i.e. the y-coordinates in the figures, are found by first sorting the constituencies in ascending order and then calculating the values for constituency $s = 1$ to m as:

(5.8) $$\text{Cumulative value for constituency } s = \frac{s}{m}$$

The formula is $\frac{m + 1 - s}{m}$ if the constituencies are sorted in descending order.

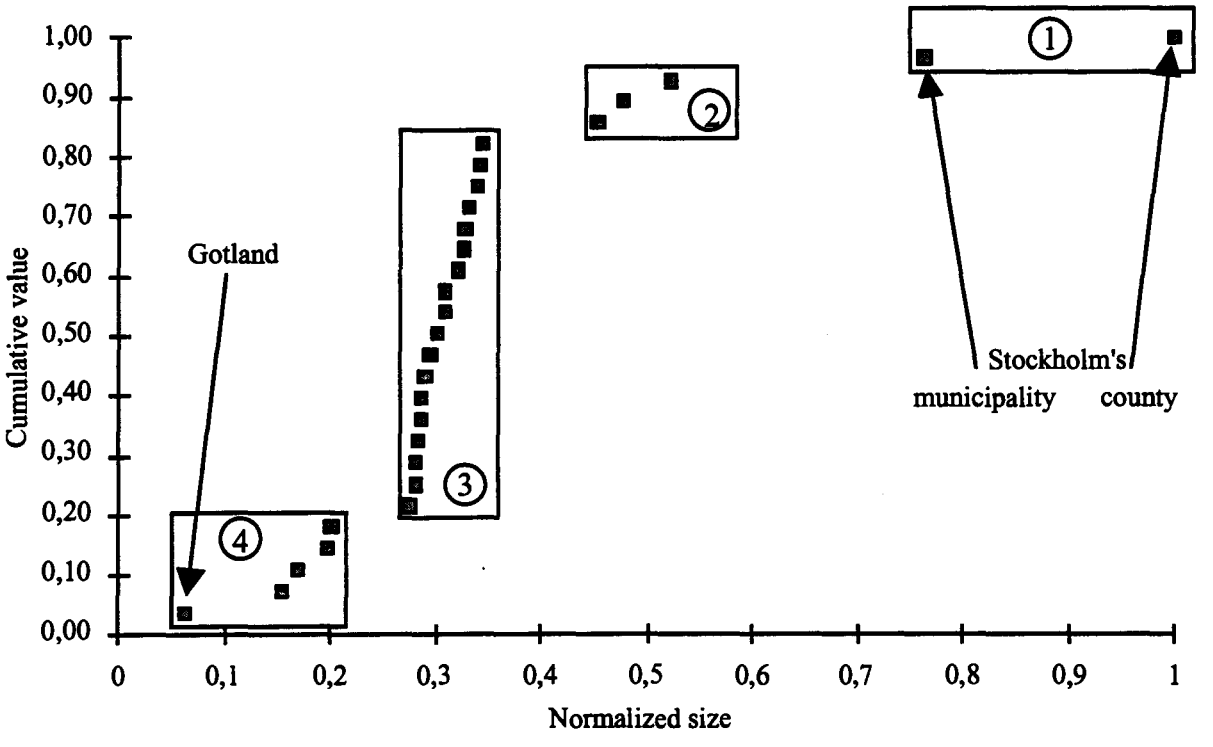
Each of the rectangular boxes in Figure 5.1 and Figure 5.2 encloses a cluster of constituencies. The encircled number inside such a box is the cluster (group) number. We treat cluster division in section 5.6.

Figure 5.1



Distribution of US state populations in 1990 and the optimal division in 3 clusters.

Figure 5.2



Distribution of Swedish constituency populations in 1990 and the optimal division in 4 clusters.

With (0,0) included as the first point, we have tried to estimate simple functional expressions for the different countries' "cumulative distribution functions". These attempts were not successful because they resulted in very large residuals for points near both ends, i.e. for (0,0), (1,1), and their neighbouring points.

Figure 5.1 and Figure 5.2 visualize the great difference between different population distributions. We categorize the US distribution as bowed and the Swedish distribution as S-shaped. To get an impression of the other population distributions without drawing figures we have calculated some descriptive statistics regarding the normalized sizes. These statistics are presented in Table 5.1 below. The column with the heading "% above mean" shows the percentage of constituencies with a normalized size above the mean.

Table 5.1

Country	*m*	Mean	% above mean	Minimum	1st Quartile	Median	3rd Quartile
Austria	9	0,575	44,44%	0,187	0,316	0,417	0,898
Denmark	17	0,506	35,29%	0,076	0,360	0,433	0,558
Norway	19	0,473	47,37%	0,156	0,304	0,424	0,527
Finland	15	0,456	46,67%	0,033	0,317	0,440	0,581
Japan	130	0,429	41,54%	0,071	0,285	0,391	0,538
Sweden	28	0,341	25,00%	0,065	0,281	0,305	0,339
Iceland	8	0,318	25,00%	0,101	0,127	0,175	0,345
Germany	16	0,288	31,25%	0,040	0,139	0,182	0,356
Canada	12	0,235	33,33%	0,003	0,053	0,105	0,274
Thailand	71	0,226	36,62%	0,028	0,114	0,160	0,284
Switzerland	26	0,224	30,77%	0,012	0,057	0,173	0,267
USA	50	0,167	30,00%	0,015	0,044	0,114	0,198
Greece	56	0,159	33,93%	0,029	0,087	0,124	0,196

Descriptive statistics regarding normalized sizes of the constituencies within 13 countries.

We categorize the Norwegian population distribution as linear, although it has a little bow at the end. The Norwegian distribution is illustrated in Figure 5.3 in section 5.7. With the US, Sweden, and Norway as references, we classify the countries in Table 5.1 as follows: Iceland, Canada, Germany, Switzerland, USA, Greece, and Thailand have population distributions which are bowed. The remaining countries, Norway, Austria, Finland, Denmark, Japan, and Sweden, have distributions ranging from linear to S-shaped. We observe that the medians for these countries are much higher than the ones for countries with bowed

distributions.

5.5 Size division

Why not utilize the normalized sizes to group the constituencies? A natural way is to let each group cover $\frac{1}{c}$ of the interval between 0 and 1. We call this method **size division**. The number of constituencies in each group is determined as follows:

Algorithm 5.3
Calculate the normalized size of every constituency by (5.7). For group $g = 1$ to c let n_g be the number of constituencies with a normalized size in the interval:

(5.9) $$\langle \frac{c-g}{c}, \frac{c-(g-1)}{c}]$$

Since the normalized sizes of all constituencies have to be calculated, size division requires more work than number division, but much less than quota division. Notice that size division is heavily influenced by the population of the most populous constituency via equation (5.7). The most important drawback of size division is that it may result in empty groups; only $g = 1$ is a guaranteed non-empty group. [B&Y] (page 128) present an example of "absolute size divison", i.e. the different groups are given in terms of quota sizes.

Number, quota, and size division have the common weak side that they do not pay enough attention to the population distribution. A consequence is that they may separate constituencies of almost equal size, thereby violating guideline 1). We continue our search for a division method which satisfies this guideline in the next section.

5.6 Cluster division

Cluster division makes the groups as homogeneous as possible internally. The different groups are thereby made as heterogeneous as possible. To determine the optimal division we minimize the following objective function, [Sp] (page 17):

$$\min D(C) = \sum_{g \in C} e(g) \tag{5.10}$$

where $e(g)$ is the sum of squared deviations within cluster (group) g:

$$e(g) = \sum_{i \in G} [p_i - \bar{p}(g)]^2 \tag{5.11}$$

and $\bar{p}(g)$ is the average population of cluster g:

$$\bar{p}(g) = \frac{1}{n_g} \cdot \sum_{i \in G} p_i \tag{5.12}$$

Because quotas and populations are proportional, defining $e(g)$ as $\sum_{i \in G} [q_i - \bar{q}(g)]^2$ will result in the same optimal division. Since the number of groups c is as small as 4 or lower in the test in section 5.8, the optimal cluster divisions were determined by complete enumeration. The investigation of the $\binom{m-1}{c-1}$ different group combinations was carried out with a Pascal program. This program has later been refined and is now a part of the MatrixBias program in Appendix 2. During enumeration it terminates calculations for inferior combinations.

For examples of optimal cluster divisions see Figure 5.1 and Figure 5.2 in section 5.4. Sometimes clusters (groups) with few members are produced. If this happens to the last group, division guideline 2) is usually violated. Such violations lead to very volatile seat-quota ratios, see section 5.8 for the definition of seat-quota ratio. We show what may happen with an example from Finland: When the 15 Finnish constituencies are divided in 4 clusters (groups), Åland becomes the only constituency in the last group. Åland is apportioned 0 seats with HA, which gives a bias of -100% in comparisons with the three other groups. Even if this example is rather extreme, it illustrates why violations of division guideline 2) should not

be tolerated.

At the end of section 5.1 we presented 2 guidelines we wanted a division to satisfy. We have no guarantee that any of the presented division methods will produce divisions which satisfy both. However, cluster division satisfies guideline 1) and quota division satisfies guideline 2). We therefore rate these division methods above size and number division. Since cluster and quota division together satisfy both guidelines, it is natural to combine their strong points. A possibility is to use cluster division and its objective function as the basis and take precaution against its shortcomings by introducing a lower bound for group quota. The formulation of a combined cluster/quota division might look like this:

$$\min D(C) = \sum_{g \in C} e(g) \tag{5.13}$$

subject to the constraint

$$\frac{3}{2} \cdot \bar{q} \geq q(g) \geq \frac{1}{2} \cdot \bar{q} \qquad \forall g \tag{5.14}$$

One could also consider introducing elements from size and number division as additional constraints. We have not tested any kind of combined division method. Let us evaluate size and number division when they stand alone: Size division takes relative size into consideration, but it leads to too many instances of empty groups. To make size division workable some subjective judgement regarding the placing of dividing lines must be applied. Number division is a mechanical rule which does not pay any attention to the characteristics of the population distribution. It satisfies division guideline 1) better for linear distributions than for S-shaped and especially bowed distributions. The best thing which can be said about number division is that it requires little work.

5.7 Illustration of the division methods

In this section we show how the four division methods would have divided the Norwegian constituencies (counties) in 1989. We have chosen to divide the 19 constituencies in three groups, i.e. $c = 3$. The basis for the divisions is the number of eligible voters in the constituencies. These and some other relevant data are presented in Table 5.2:

Table 5.2

County	Eligible voters	Quota	Normalized size
Oslo	354750	18,35	1,000
Akershus	307608	15,91	0,867
Hordaland	303618	15,70	0,856
Rogaland	237729	12,30	0,670
Sør-Trøndelag	191015	9,88	0,538
Østfold	183192	9,47	0,516
Nordland	182754	9,45	0,515
Møre og Romsdal	178764	9,25	0,504
Buskerud	170429	8,81	0,480
Vestfold	150323	7,77	0,424
Hedmark	146215	7,56	0,412
Oppland	141643	7,33	0,399
Telemark	124944	6,46	0,352
Troms	111295	5,76	0,314
Vest-Agder	104443	5,40	0,294
Nord-Trøndelag	95505	4,94	0,269
Sogn og Fjordane	79484	4,11	0,224
Aust-Agder	71123	3,68	0,200
Finnmark	55477	2,87	0,156
SUM	3190311	165,00	

Eligible voters, quotas with a house size of 165, and normalized sizes for the 19 Norwegian constituencies in 1989.

We start with number division. The figures $\overline{m} = 19 \text{ div } 3 = 6$ and $\overline{r} = 19 \bmod 3 = 1$ mean that the last group contains $6 + 1 = 7$ constituencies, while the two other groups have 6 each. Hence, the number division is $\mathbf{n} = (6, 6, 7)$. By summing the quotas of constituencies which belong to the same group, we find that the quota sizes of the groups are 81,61, 50,17, and 33,22 respectively.

The size division is found by inspection of Table 5.2. By counting the number of constituencies with a normalized size higher than $\frac{2}{3}$, we find that 4 constituencies belong to the first group. There are 6 constituencies with a normalized size of $\frac{1}{3}$ or lower. They belong to the last group. The remaining 9 constituencies belong to the middle (second) group. Thus, the size division is $\mathbf{n} = (4, 9, 6)$. These groups have quota sizes of 62,26, 75,98, and 26,76 respectively.

The average quota per group $\bar{q}$ is a central figure when quota division is applied. Here $\bar{q} = \frac{165}{3} = 55$. We use Algorithm 5.2 to generate candidates for the different groups. From the size division above we know that the total quota of the four largest constituencies is $q_{(1,4)} = 62,26$. This total is higher than the average group quota $\bar{q}$, so the total quota of the three largest constituencies must be calculated. It is $q_{(1,3)} = 49,96$, i.e. lower than $\bar{q}$. Thus, $z = 3$, which means that the upper bound is set equal to the quota of the fourth largest constituency, i.e. $\varpi = q_4 = 12,30$. We move on to the generation of candidates for $g = 1$. Because $|q_{(1,5)} - \bar{q}| = |72,14 - 55| > \varpi$, only $(x,y) = (1,3)$ and $(1,4)$ are candidates for the first group. The absolute deviations from the average group quota for these two candidates are $|q_{(1,3)} - \bar{q}| = 5,04$ and $|q_{(1,4)} - \bar{q}| = 7,26$ respectively.

The next task is generation of candidates for $g = 2$. Let us start by generating candidates connected to $(x,y) = (1,3)$. We calculate $q_{(4,y)}$ for different choices of y and find that $|q_{(4,y)} - \bar{q}| \leq \varpi$ for $(x,y) = (4,8)$, $(4,9)$, and $(4,10)$. The quota sizes of these candidates are $q_{(4,8)} = 50,35$, $q_{(4,9)} = 59,16$, and $q_{(4,10)} = 66,93$, with absolute deviations of 4,65, 4,16, and 11,93 respectively. Similarly, candidates connected to $(x,y) = (1,4)$ are generated. Their quota sizes are $q_{(5,9)} = 46,86$, $q_{(5,10)} = 54,63$, and $q_{(5,11)} = 62,19$ with absolute deviations of 8,14, 0,37, and 7,19 respectively.

Now it is time to generate candidates for the third group. These candidates must be connected to a candidate for the second group and include the smallest constituency. We calculate the total quota for possible candidates: $q_{(9,19)} = 64{,}69$, $q_{(10,19)} = 55{,}88$, and $q_{(11,19)} = 48{,}11$, with absolute deviations of 9,69, 0,88, and 6,89 respectively. Since $q_{(12,19)} = 40{,}55 < 55 - \varpi$, $(X,Y) = (12,19)$ is not a candidate for $g = 3$. Moreover, this means that the connected candidate for the second group $(x,y) = (5,11)$ must be eliminated. The candidates which are left after the elimination process are summarized in Table 5.3:

Table 5.3

s	(x,y)	Absolute deviation from $\bar{q}$	Candidate for	A	B	C	D	E
1	(1,3)	5,04	$g = 1$	*	*	*		
2	(1,4)	7,26	$g = 1$				#	#
3	(4,8)	4,65	$g = 2$	*				
4	(4,9)	4,16	$g = 2$		*			
5	(4,10)	11,93	$g = 2$			*		
6	(5,9)	8,14	$g = 2$				#	
7	(5,10)	0,37	$g = 2$					#
8	(9,19)	9,69	$g = 3$	*				
9	(10,19)	0,88	$g = 3$		*		#	
10	(11,19)	6,89	$g = 3$			*		#

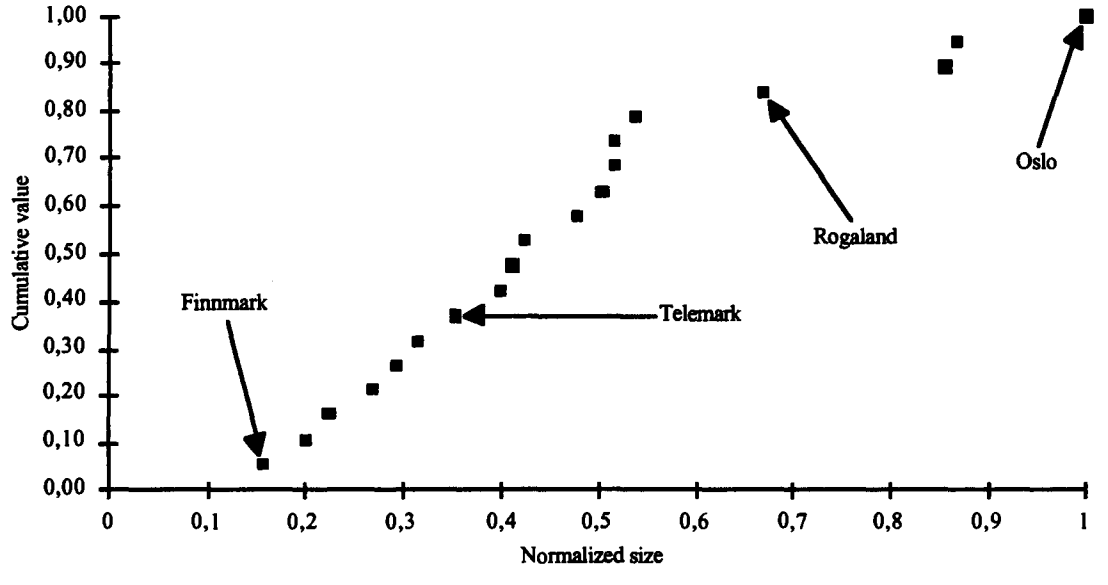
Candidates for groups, their absolute deviation, and the combination of them into connected trees of groups (A - E).

Below we use the words "group combination" instead of the term "connected tree of groups". Table 5.3 reveals that there are 5 group combinations with a maximal absolute deviation of $\varpi = 12{,}30$ or less. The maximal absolute deviations are 9,69 for group combination A, 5,04 for B, 11,93 for C, 8,14 for D, and 7,26 for E. Thus, B is the optimal group combination, which means that the quota division is $\mathbf{n} = (3, 6, 10)$. The quota sizes of these groups are 49,96, 59,16, and 55,88 respectively.

A reason why the number of "interesting" group combinations is as small as 5 here is the small number of groups c. We observe that the number of candidates for the second group is equal to the total number of candidates for the end groups ($g = 1$ and $g = 3$). Candidates for the end groups are restricted on one side, while candidates for the middle group ($g = 2$) are freer. We expect the number of group combinations with a maximal absolute deviation not larger than ϖ to increase sharply for problems with a higher number of groups.

Before we present the cluster division, let us take a look at a graphical illustration of the population distribution:

Figure 5.3



Distribution of Norwegian constituency populations in 1989.

We calculated D(C) for all $\binom{19-1}{3-1} = 153$ group combinations to determine the optimal cluster division. The optimal cluster division is $\mathbf{n} = (3, 9, 7)$, which means that group 1 starts with Oslo, group 2 with Rogaland and group 3 with Telemark, see Figure 5.3. The quota sizes of these groups are 49,96, 81,82, and 33,22 respectively. We illustrate the calculations which have been carried out for every group combination by presenting calculations for the optimal combination: For each group we calculate the average quota and the sum of squared deviations,

which for group 1 are:

(5.15)
$$q(1) = \frac{18{,}35 + 15{,}91 + 15{,}70}{3} \approx 16{,}65$$
$$e(1) = (18{,}35\text{-}16{,}65)^2 + (15{,}91\text{-}16{,}65)^2 + (15{,}70\text{-}16{,}65)^2 \approx 4{,}34$$

Similar calculations for groups 2 and 3 give $q(2) \approx 9{,}09$, $e(2) \approx 18{,}49$, $q(3) \approx 4{,}75$, and $e(3) \approx 9{,}49$. The value of the objective function then becomes:

(5.16)
$$D(C) = e(1) + e(2) + e(3) \approx 4{,}34 + 18{,}49 + 9{,}49 = 32{,}32$$

For quota division we were able to restrict the number of group combinations which had to be evaluated. It should be possible to reduce the workload for cluster division too. A possibility is to use visual inspection of the "cumulative distribution function" to find a "good" group combination. The value of the objective function for this group combination can then be used as an initial upper bound.

5.8 Bias test

Bias is the systematic deviation between the apportionment to subsets of constituencies and the quota (fair share) of these constituencies. Thus, bias is related to the apportionment method employed. The deviation between apportionment and quota varies from one election situation to the next, so we need observations from several situations to draw general conclusions. It is unsuitable to let each constituency be its own group, so we divide the constituencies in a small number of groups. As will be seen from the figures later in the section, division methods have an impact on the reported biases through their selection of constituencies. A natural bias measure for a group g is the **seat-quota ratio**:

(5.17)
$$k_g = \frac{\sum_{i \in G} a_i}{\sum_{i \in G} q_i}$$

k_g is a kind of bias index, with $k_g = 1$ as the unbiased point. When $k_g > 1$, the constituencies in group g have got more seats than they deserve, while $k_g < 1$ means that the group has been apportioned fewer seats than its fair share. The percentage bias of the apportionment to the constituencies in g is $(k_g - 1) \cdot 100\%$. This percentage measures the bias for this particular group. Sometimes it is more interesting to compare two groups directly. This is what we do in our bias test. In this test we employ a modified version of the empirical measure from [B&Y] (page 126). Let D and E be two disjoint groups of constituencies $D \cap E = \varnothing$, where neither is empty and the constituencies in group D are larger than the ones in E. We define the bias between these two groups as:

$$\varepsilon_{DE} = \frac{k_D - k_E}{k_D} \tag{5.18}$$

We calculate the bias towards the D group instead of the bias against the D group as [B&Y] do. This implies a sign shift compared with their formula. Another difference is that [B&Y] use p_i in place of q_i in (5.17). q_i is equal to p_i multiplied by the constant $\frac{h}{p_M}$, so the absolute value of our bias percentage ε_{DE} is equal to the absolute value of theirs. The advantage of using q_i is an interpretable k_g, because its value is in the neighbourhood of 1. When ε_{DE} is positive, group D is favoured compared with group E. A negative ε_{DE} means that the favouritism goes the other way. Let a be the country (observation) index, n the number of countries (observations), and ε_{DE_a} the observed bias between group D and E for country a. The mean of the bias observations, denoted $\bar{\varepsilon}_{DE}$, is our estimate of the bias percentage:

$$\bar{\varepsilon}_{DE} = \frac{1}{n} \cdot \sum_{a=1}^{n} \varepsilon_{DE_a} \tag{5.19}$$

The number of pairwise bias comparisons depends on the number of groups through the formula $\binom{c}{2}$. We carry out tests with the constituencies divided in 2, 3, and 4 groups, which result in 1, 3, and 6 comparisons (group pairs) respectively. Bias percentages are estimated for every group pair. We assume that all bias observations ε_{DE_a} for a particular group pair are normally distributed with the same mean and variance. With the additional assumption of independence

we can apply the t-test to draw conclusions about bias directions of different apportionment methods. Since the population distributions in our test differ substantially, there may be a question mark with the assumption that all observations of ε_{DE} are from the same normal distribution. However, based on a brief inspection of bias observations, the normality assumption does not seem unreasonable.

We call a test where all observations are from the same country a country test. The results of a country test are of interest in the determination of which apportionment method to apply in this country. Because successive election situations within a country are correlated, we suspect that successive bias observations are correlated too. If this is the case, the independence assumption is doubtful for country tests. We have neither carried out country tests regarding bias nor checked the correlation between successive bias observations.

We return to our bias test. The null hypothesis for every apportionment method is that the method is unbiased:

(5.20) $\quad H_0: \varepsilon_{DE} = 0$

The alternative hypotheses for the different apportionment methods build on the theoretical framework from chapter 4. Both MF and LF are approximately coalition-neutral, which is a theoretical property. The alternative hypothesis for them is that they are biased in one direction or the other, which implies a two-sided test. The tests for the remaining methods are one-sided. Supposedly, MF is the middle point among the divisor methods. Then it is reasonable to let the alternative hypotheses for the other divisor methods be as follows: HA is biased towards the group consisting of the larger constituencies in the group pair, while SD, HM, and EP are biased towards the group consisting of the smaller constituencies. A summary of the alternative hypotheses:

(5.21) H_A: $\varepsilon_{DE} \neq 0$ for MF, LF
$\varepsilon_{DE} > 0$ for HA
$\varepsilon_{DE} < 0$ for SD, HM, EP

We use T_ε as our test statistic, see [Li] (page 265-267):

$$(5.22) \qquad T_\varepsilon = \frac{\bar{\varepsilon}_{DE} - 0}{\frac{s_\varepsilon}{\sqrt{n}}}$$

where $s_\varepsilon = \sqrt{s_\varepsilon^2}$ is the sample standard deviation and s_ε^2 is calculated as:

$$(5.23) \qquad s_\varepsilon^2 = \frac{1}{n-1} \cdot \sum_{a=1}^{n} (\varepsilon_{DE_a} - \bar{\varepsilon}_{DE})^2$$

We calculate the test statistic T_ε for every estimated bias percentage. Student's t-distribution with n - 1 degrees of freedom is used to draw conclusions about the statistical significance of the estimates. We use a level of significance of 5% in the test. How the four division methods divide the different countries in 2, 3, and 4 groups is shown in Appendix 1. Appendix 1 also presents estimated bias percentages, standard errors of these estimates, t-statistics, and P-values for the different apportionment methods given these divisions. The estimated bias percentages with division in three groups, i.e. with c = 3, and the statistical significance of these estimates are shown in Table 5.4:

Table 5.4

Division	$\bar{\varepsilon}$	SD	HM	EP	MF	HA	LF
Number	(L,M)	-4,08% **	-0,75%	-0,60%	0,17%	4,07% **	0,12%
	(L,S)	-14,36% **	-5,67% *	-3,91% *	1,07%	13,73% **	1,95% *
	(M,S)	-9,81% **	-4,87% *	-3,29% *	0,86%	10,14% **	1,78%
Quota	(L,M)	-3,47% **	-0,63%	-0,52%	-0,42%	2,11% **	-0,57%
	(L,S)	-9,11% **	-2,12% *	-1,44%	0,59%	8,53% **	0,58% *
	(M,S)	-5,39% **	-1,47% *	-0,90%	1,00% *	6,60% **	1,14% **
Size	(L,M)	-2,36% **	-0,35%	0,03%	0,32%	2,27% **	0,29%
	(L,S)	-9,57% **	-2,36% *	-1,59% *	-0,06%	8,38% **	-0,10%
	(M,S)	-6,99% **	-1,99% *	-1,63% *	-0,40%	6,28% **	-0,41%
Cluster	(L,M)	-2,92% **	-0,22%	-0,09%	0,06%	2,48% **	0,01%
	(L,S)	-9,80% **	-2,58% *	-1,73% *	0,15%	9,32% **	0,32%
	(M,S)	-6,62% **	-2,36% *	-1,66%	0,06%	7,06% **	0,29%

Estimated bias percentages with division in 3 groups: Large (L), Medium (M), and Small (S).
Two-sided test for MF and LF, one-sided for the other apportionment methods.
** = Significant at the 1%-level * = Significant at the 5%-level

We observe that the bias percentages are increasing from left to right in Table 5.4, with the exception of LF which is not a divisor method. This confirms the theoretical result SD > HM > EP > MF > HA deduced from Theorem 4.3. With a few exceptions, the absolute values of the bias percentages with number division are larger than the corresponding bias percentages with the other division methods. Quota, size, and cluster division result in about the same bias percentages. The estimates with size division are based on 12 countries, because the middle group was empty for Canada when divided in three groups. As mentioned in section 5.6, cluster division is unsatisfactory if the last group consists of constituencies which in total are apportioned no or very few seats. The result of such unsatisfactory cluster divisions can be seen in Appendix 1, where the estimated bias percentages for HA with $g = 4$ are very large for comparisons which involve group 4.

What conclusions regarding bias directions can be drawn from the data in Table 5.4? All estimates for SD and HA are significant at the 1%-level. SD is biased towards the group of smaller constituencies in the group pair, while HA is biased towards the group of larger constituencies. Regarding HM, all estimated bias percentages for group pairs which involve the smallest group S are significant at the 5%-level. Thus, HM is biased towards S. The same may be said for EP, but in this case only 5 out of 8 estimates are significant at the 5%-level. Are MF and LF unbiased? The estimated bias percentages suggest that they both are slightly biased towards the group of larger constituencies in the group pair. However, only a few of these estimates are statistically significant, so we do not reject the null hypothesis that MF and LF are unbiased.

Let us compare our estimates with the bias percentages in [B&Y] (page 126). Their findings are based on 19 apportionments of the House of Representatives in the US until 1980, where states (constituencies) with a quota less than 0,5 are eliminated. [B&Y] apply their version of number division and present bias percentages for the comparison between the group of the largest and the group of the smallest states. We compare these percentages with our estimates $\bar{\varepsilon}_{(L,S)}$ with number division. With the exception of MF, the bias directions are the same. The

absolute values of our bias percentages for SD and HA are smaller than theirs, while for HM, EP, and MF ours are higher. [B&Y] do not calculate the bias percentage for LF.

To round off this section we deduce how the third pairwise bias percentage is determined by the two others when $c = 3$. We assume that $k_L > 0$ and $k_M > 0$ in this deduction, i.e. that both L and M are awarded seats. From (5.18) we find the formulas for the biases in Table 5.4:

(5.24) $$\varepsilon_{(L,M)} = \frac{k_L - k_M}{k_L} \qquad \varepsilon_{(L,S)} = \frac{k_L - k_S}{k_L} \qquad \varepsilon_{(M,S)} = \frac{k_M - k_S}{k_M}$$

The difference between $\varepsilon_{(L,S)}$ and $\varepsilon_{(L,M)}$ can be expressed as:

(5.25) $$\varepsilon_{(L,S)} - \varepsilon_{(L,M)} = \frac{k_M - k_S}{k_L}$$

The numerator here is the same as in the expression for $\varepsilon_{(M,S)}$ in (5.24). $\varepsilon_{(M,S)}$ can therefore be expressed as:

(5.26) $$\varepsilon_{(M,S)} = \frac{k_L}{k_M} \cdot [\varepsilon_{(L,S)} - \varepsilon_{(L,M)}]$$

To get rid of k_L (and k_M), we deduce an expression for k_L from the formula for $\varepsilon_{(L,M)}$ in (5.24):

(5.27) $$k_L = \frac{k_M}{1 - \varepsilon_{(L,M)}}$$

By putting this expression for k_L into (5.26) and simplifying, we get:

(5.28) $$\varepsilon_{(M,S)} = \frac{\varepsilon_{(L,S)} - \varepsilon_{(L,M)}}{1 - \varepsilon_{(L,M)}} \qquad \text{for } k_L, k_M > 0$$

which shows that $\varepsilon_{(M,S)}$ is fully determined by the two other bias percentages.

Chapter 6: Formulations

The normal formulations of the different apportionment methods were presented in chapters 1 and 2. In this chapter we show how the apportionment methods can be formulated as constrained optimization problems. What distinguish the methods from each other are the objective functions. During the first 8 sections we investigate which objective functions that lead to the different apportionment methods. The common constraints for all these formulations are the basic ones, namely that the variables a_i are non-negative integers which sum to h:

(6.1) $\quad a_i \in N_0 \qquad$ for all i

(6.2) $\quad \sum_{i \in M} a_i = h$

The last two sections present formulations with a somewhat different perspective. In section 6.9 we study how divisor methods are connected with the utility concept, while formulations which compare pairs of constituencies is the topic in the last section.

6.1 The method of the largest fraction

An ideal for an apportionment is that it is close to the quotas. The absolute value of the deviation between the apportionment and the quota $|a_i - q_i|$ can be used to measure this closeness. Let us make the largest such deviation as small as possible:

(6.3) $\quad \min_{\mathbf{a}} \max_{i \in M} |a_i - q_i|$

It is clear from step 2 of Algorithm 1.1 that this is exactly what LF does, so the objective function given by (6.3) results in the LF apportionment. What happens if the sum of the deviations is minimized?

$$\min \sum_{i \in M} |a_i - q_i| \tag{6.4}$$

(6.4) also yields LF. Consider the following objective function:

$$\min \sum_{i \in M} |a_i - q_i|^p \qquad \text{where } p \geq 1 \text{ and real} \tag{6.5}$$

The p used here must not be confused with the notation of population **p**. [H-A] (page 10-13) treats the close relations between the objective function in (6.5) and what is known as the l_p**-norm** (and l_∞-norm). The higher p is, the more weight is given to large deviations. However, (6.5) characterizes LF for all $p \geq 1$, even for $p = \infty$ with some minor modifications. The l_p-norm is also encountered in connection with controlled rounding in chapter 15.

6.2 An attempt for Lowndes' method

It is tempting to try the following objective function for Lowndes' method:

$$\min_{\mathbf{a}} \max_{i \in M} \frac{q_i - a_i}{\lfloor q_i \rfloor} \tag{6.6}$$

To follow up the suggested definition from section 1.2, we define $\frac{q_i}{\lfloor q_i \rfloor} > \frac{q_s}{\lfloor q_s \rfloor} \geq 1$ if $0 < q_s < q_i < 1$ and $\frac{q_i - 1}{\lfloor q_i \rfloor} < 0$ if $0 < q_i < 1$. (6.6) works well if the quota of every constituency is at least 1. However, it may run into problems if there are two or more constituencies with quotas smaller than 1. Consider the quota vector $\mathbf{q} = (2\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$. According to (6.6), the optimal apportionment for this election situation is $\mathbf{a} = (1, 1, 1)$. Lowndes' method would have distributed 2 seats to the first constituency, so the objective function given by (6.6) does not imply Lowndes' method.

6.3 Attempts for the harmonic mean method

Neither the objective function in (6.7) nor in (6.8) gives HM. For an example of their failure see [B] (page 137).

(6.7) $$\min \sum_{i \in M} \left| \frac{p_i}{a_i} - \frac{p_M}{h} \right|$$

(6.8) $$\min_{\mathbf{a}} \max_{i \in M} \left| \frac{p_i}{a_i} - \frac{p_M}{h} \right|$$

However, [E] (page 1216) states that if HM produces an apportionment which stays within quota, then this apportionment minimizes (6.7) among all apportionments which stay within the quota. Since HM does violate staying within the quota for some election situations, this result is not very useful from a practical viewpoint.

6.4 The method of the highest average

An important ratio when comparing apportionments for different constituencies is the **average number of seats per individual for a constituency** $\frac{a_i}{p_i}$. The reciprocal of this ratio, i.e. the **average number of people per seat for a constituency** $\frac{p_i}{a_i}$, may also be used. The greater the former ratio is, the better the constituency has fared in the apportionment. One way of utilizing the ratio is to require that the most favoured constituency is as little advantaged as possible. Mathematically:

(6.9) $$\min_{\mathbf{a}} \max_{i \in M} \frac{a_i}{p_i}$$

(6.9) gives HA as do the objective function: $\max_{\mathbf{a}} \min_{i \in M} \frac{p_i}{a_i}$.

6.5 The method of the smallest divisor

Another way of utilizing the ratio $\frac{a_i}{p_i}$ is to make the least favoured constituency as advantaged as possible:

(6.10) $$\max_{\mathbf{a}} \min_{i \in M} \frac{a_i}{p_i}$$

(6.10) yields SD. Another objective function which implies SD is: $\min_{\mathbf{a}} \max_{i \in M} \frac{p_i}{a_i}$. Notice that (6.10) is the mirror image of (6.9).

6.6 The method of major fractions

When the apportionment to a constituency deviates from the quota, i.e. $a_i \neq q_i$, the average number of seats per individual for this constituency $\frac{a_i}{p_i}$ deviates from the **national average number of seats per individual** $\frac{h}{p_M}$. By squaring this deviation, i.e. $(\frac{a_i}{p_i} - \frac{h}{p_M})^2$, positive and negative deviations are treated equally and large deviations are given more weight than small ones. In 1910 André Sainte-Laguë argued that the individuals are the basic elements which shares should be made as equal as possible, see [L]. He therefore weighted the squared deviation for a constituency by the number of people in the constituency. Minimization of the sum of such terms over all constituencies led him to the objective function:

(6.11) $$\min \sum_{i \in M} [p_i \cdot (\frac{a_i}{p_i} - \frac{h}{p_M})^2]$$

We simplify and get: $\min \sum_{i \in M} \frac{(a_i)^2}{p_i} - \frac{h^2}{p_M}$. The last term is a constant, so (6.11) is equivalent to:

(6.12) $$\min \sum_{i \in M} \frac{(a_i)^2}{p_i}$$

[B&Y] (page 103-104) prove that this objective function implies MF.

6.7 The method of equal proportions

Rather than making the shares of the individuals as equal as possible, we could try to make the average number of people per seat for the different constituencies as equal as possible. To accomplish this we use the squared deviation $(\frac{p_i}{a_i} - \frac{p_M}{h})^2$ with a_i as weight and sum over all constituencies:

(6.13) $$\min \sum_{i \in M} [a_i \cdot (\frac{p_i}{a_i} - \frac{p_M}{h})^2]$$

This objective function can be simplified to: $\min \sum_{i \in M} \frac{(p_i)^2}{a_i} - \frac{(p_M)^2}{h}$, which further is equivalent to: $\min \sum_{i \in M} \frac{(p_i)^2}{a_i}$. By using the same approach as [B&Y] apply for MF, it can be proved that (6.13) results in EP.

6.8 The constant parametric divisor methods

[O] (page 205) gives the result that a constant parametric divisor method minimizes the objective function:

(6.14) $$\min \sum_{i \in M} [p_i \cdot (\frac{a_i + t - \frac{1}{2}}{p_i} - \frac{h}{p_M})^2]$$

He shows that (6.14) is equivalent to: $\min \sum_{i \in M} \frac{(a_i + t - \frac{1}{2})^2}{p_i}$ and proves, in the same way as [B&Y] do for MF, that this objective function implies the constant parametric divisor method with parameter t.

6.9 Maximization of utility

Utility is an important concept in welfare economics. In this section we relate it to the apportionment of seats. We make the following assumptions regarding the relationship between number of representatives and utility:

The utility an individual in constituency i derives from having l representatives to represent his/her constituency is denoted $u_i(l)$. In particular, the utility of being without any representative is zero, i.e. $u_i(0) = 0$. All people in the country derive the same utility from being represented by l representatives:

(6.15) $$u_i(l) = u_s(l) \quad \text{for all } i, s \in M$$

When there is no need to know the specific constituency involved, we write $u(l)$ instead of $u_i(l)$. The total utility for the whole country is found by adding the utilities of all people. Our typical individual wants to be represented by as many representatives as possible. This is the normal assumption for utility functions that more is preferred to less. It means that the utility function is an increasing function of the number of representatives:

(6.16) $$u(l - 1) < u(l)$$

The first seat won is more precious than the second seat won etc. Hence, the utility gained by increasing the representation from l to $l + 1$ is always less than the utility gained by increasing the representation from $l - 1$ to l:

(6.17) $$u(l + 1) - u(l) < u(l) - u(l - 1)$$

This is the assumption of diminishing marginal utility, i.e. the utility function is a strictly concave function. Our utility function is only defined for non-negative integers, so it is discontinuous. An alternative to (6.17) is to assume that the utility gained from getting the $(l + 1)$th seat is never greater then the utility

gained from getting the lth seat, i.e. the substitution of $\leq$ for $<$ in (6.17). With this alternative assumption the utility function would have been concave. But we stick to the assumption made by (6.17).

We call the increase in utility from $u(l - 1)$ to $u(l)$ utility increment and denote it u_l:

(6.18) $$u_l = u(l) - u(l - 1)$$

It follows from (6.16) that u_l is positive for all l. Furthermore, (6.17) tells us that the utility increments form a decreasing sequence in l. When it is of interest to know which constituency a utility increment is associated with, we denote it u_{il}.

We want to apportion seats among constituencies in such a way that the national utility is maximized subject to constraints (6.1) and (6.2):

(6.19) $$\max \sum_{i \in M} [p_i \cdot u_i(l)]$$

[H-A] (page 65) proves that this utility maximizing approach implies that a strict divisor method in the sense of Definition 2.1 is being used. The divisors in such a method are given as:

(6.20) $$d_l = \frac{1}{u_l}$$

which means that the relationship between utility function and divisor series is:

(6.21) $$u(0) = 0$$
$$u(s) = \sum_{l=1}^{s} \frac{1}{d_l} \qquad s \in H$$

We introduce a_{il} as a 0/1-variable and define it as follows: It is 1 if constituency i has got its lth seat and 0 if not. Notice that this definition means that $a_{i(l+1)} = 1$ implies $a_{il} = 1$. By combining the definition of a_{il} with (6.20), the total utility gained by constituency i getting its lth seat can be written as:

(6.22) $$p_i \cdot u_{il} = \frac{p_i}{d_l} = \left(\frac{p_i}{d_l}\right) \cdot a_{il}$$

The national utility is therefore maximized by giving out one seat at a time to the constituency which gains most by getting the next seat. This is exactly the same procedure as for the distribution of seats with a divisor method, see Algorithm 2.1.

We assume that $m \leq h$ and permit division by 0 the same way as in Definition 2.1. With this in hand we present a constrained optimization formulation of strict divisor methods:

(6.23) $$\max \sum_{i \in M} \sum_{l \in H} \left[\left(\frac{p_i}{d_l}\right) \cdot a_{il}\right]$$

subject to the constraints

(6.24) $$a_{il} = 0 \text{ or } 1 \qquad \text{for all } i \in M, l \in H$$

(6.25) $$\sum_{i \in M} \sum_{l \in H} a_{il} = h$$

This way of formulating the apportionment problem will be important when we reach the matrix apportionment problem in part II.

6.10 Pairwise comparisons

Pairwise comparisons of constituencies is one way of measuring the inequality an apportionment has created. The difference between two terms which express how two constituencies have fared is called **amount of inequality**. To compare how two constituencies i and s have fared in the apportionment we could look at the absolute difference between the number of seats per individual ratios, i.e. $\frac{a_i}{p_i} - \frac{a_s}{p_s}$, or the relative difference between these ratios $[\frac{a_i}{p_i} - \frac{a_s}{p_s}] / \frac{a_s}{p_s}$. Furthermore, these differences can be rearranged by crossmultiplying them in many ways. For instance, crossmultiplying $\frac{a_i}{p_i} - \frac{a_s}{p_s}$ by p_i yields $(a_i - p_i \cdot \frac{a_s}{p_s})$. A total of 16 different absolute differences can be found by crossmultiplication, considering only positive differences. [Hu] (page 104-108) showed that 4 of these absolute differences are unworkable, see Table 6.2, while the 12 remaining imply one of the five historical divisor methods, see Table 6.1. Moreover, he showed that all 16 relative differences give EP, which was the main reason for his strong recommendation of this method.

The idea behind pairwise comparisons is to transfer seats from the advantaged constituency in a pair to the disadvantaged constituency until the amount of inequality is minimized for all $\binom{m}{2}$ pairs of constituencies. A transfer should be made if it leads to a decrease in the amount of inequality, and it may also be made if the amount of inequality stays the same. When the amount of inequality does not change following a transfer, we have a tie involving these two constituencies. This can be seen by considering only the equality case in Example 6.1 below. We label constituencies such that constituency i is never disfavoured in the pairwise comparison with constituency s, i.e. $\frac{a_i}{p_i} \geq \frac{a_s}{p_s}$. If the transfer of a seat from i to s reverses the sign of this inequality, we relabel the constituencies. Thus, all differences considered are non-negative; only in the case of i and s being equally well represented is the difference equal to zero.

Table 6.1

	MF	HM	SD	HA	EP
Comparison 1	$\frac{a_i}{p_i} - \frac{a_s}{p_s}$	$\frac{p_s}{a_s} - \frac{p_i}{a_i}$	$a_i - p_i \cdot \frac{a_s}{p_s}$	$p_s \cdot \frac{a_i}{p_i} - a_s$	$\frac{a_i}{p_i} \cdot \frac{p_s}{a_s} - 1$
Comparison 2	$a_i \cdot p_s - a_s \cdot p_i$	$\frac{1}{p_i \cdot a_s} - \frac{1}{p_s \cdot a_i}$	$a_i \cdot \frac{p_s}{a_s} - p_i$	$p_s - a_s \cdot \frac{p_i}{a_i}$	$1 - \frac{a_s}{p_s} \cdot \frac{p_i}{a_i}$
Comparison 3			$\frac{1}{a_s} \cdot \frac{a_i}{p_i} - \frac{1}{p_s}$	$\frac{1}{p_i} - \frac{1}{a_i} \cdot \frac{a_s}{p_s}$	

Pairwise comparisons for the five historical divisor methods.

All 12 absolute pairwise comparisons in Table 6.1 are such that we sooner or later arrive at an apportionment where further transfers of seats will not reduce the amount of inequality for any pair of constituencies.

Table 6.2

Comparisons	$\frac{a_i}{a_s} - \frac{p_i}{p_s}$	$\frac{p_s}{p_i} - \frac{a_s}{a_i}$	$\frac{1}{a_s} - \frac{1}{p_s} \cdot \frac{p_i}{a_i}$	$\frac{1}{p_i} \cdot \frac{p_s}{a_s} - \frac{1}{a_i}$

Unworkable pairwise comparisons.

[Hu] (page 104-106) presents examples which show that the 4 absolute pairwise comparisons in Table 6.2 are unworkable. For many election situations with three or more constituencies these comparisons are unable to determine which is the optimal apportionment. This happens because it is possible to reduce the amount of inequality for one of the constituency pairs by carrying out yet another transfer.

In practice it is timeconsuming to use pairwise comparisons to apportion seats. $\frac{m \cdot (m-1)}{2}$ pairs have to be investigated and for each of them at least one comparison has to be made. It is much easier to use Algorithm 2.1 for the appropriate divisor method, then only $h + m - 1$ quotients have to be calculated.

Example 6.1 demonstrates how we determine which divisor method, if any, a particular pairwise comparison implies:

Example 6.1

We focus on the pairwise comparison: $\frac{a_i}{p_i}\cdot\frac{p_s}{a_s}$ - 1. Table 6.1 tells us that this comparison implies EP. We are now going to prove this:

The starting point is the final apportionment, which is the result of all pairwise comparisons. We focus on a pair of constituencies with apportionments a_i and a_s. Because we always consider non-negative differences, constituency i is not disfavoured compared with constituency s. In the following we permit division by zero the same way as in Definition 2.1. The situation where neither i nor s has got any seat is trivial, there is no seat to transfer so the amount of inequality cannot be reduced. We therefore assume that $a_i \geq 1$ and $a_s \geq 0$. Since constituency s is not apportioned more seats, the transfer of one seat from i to s will not decrease the amount of inequality. In other words, constituency s will not be less favoured over constituency i after such a transfer than i is favoured over s now. Thus, the following inequality holds:

$$\frac{a_i}{p_i}\cdot\frac{p_s}{a_s} - 1 \leq \frac{a_s + 1}{p_s}\cdot\frac{p_i}{a_i - 1} - 1 \tag{6.26}$$

Elimination of the last terms and multiplication of both sides by $(p_i \cdot p_s)$ yields:

$$\frac{(p_s)^2\cdot a_i}{a_s} \leq \frac{(p_i)^2\cdot(a_s + 1)}{a_i - 1} \Leftrightarrow \frac{(p_s)^2}{a_s\cdot(a_s + 1)} \leq \frac{(p_i)^2}{(a_i - 1)\cdot a_i} \Leftrightarrow$$

$$\frac{p_s}{\sqrt{a_s\cdot(a_s + 1)}} \leq \frac{p_i}{\sqrt{(a_i - 1)\cdot a_i}} \tag{6.27}$$

We recognize the denominators in these expressions as the divisors $d_{(a_s+1)}$ and d_{a_i} for EP. Thus, (6.27) is the min-max inequality for EP. This proves that the pairwise comparison $\frac{a_i}{p_i}\cdot\frac{p_s}{a_s}$ - 1 implies EP. The proofs for the other workable pairwise comparisons are similar.

In the following we use the abbreviation UW for unworkable pairwise comparisons and number these comparisons from left to right in Table 6.2. Moreover, we let A_a stand for pairwise comparison number a for apportionment method A. In this connection we regard UW as an apportionment method. The number a is

taken from Table 6.1 or Table 6.2, so $p_s - a_s \cdot \frac{p_i}{a_i}$ is HA_2, $\frac{p_s}{p_i} - \frac{a_s}{a_i}$ is UW_2 etc. The populations p_i and p_s are constants in the pairwise comparisons, while a_i and a_s are variables. All pariwise comparisons can be expressed as another comparison multiplied by a constant. We have the following relationships:

(6.28)
$$\begin{aligned} MF_1 &= \frac{1}{p_i \cdot p_s} \cdot MF_2 = \frac{1}{p_i} \cdot SD_1 = \frac{1}{p_s} \cdot HA_1 \\ HM_1 &= p_i \cdot p_s \cdot HM_2 = p_s \cdot UW_3 = p_i \cdot UW_4 \\ EP_1 &= \frac{1}{p_i} \cdot SD_2 = p_s \cdot SD_3 = \frac{p_s}{p_i} \cdot UW_1 \\ EP_2 &= \frac{1}{p_s} \cdot HA_2 = p_i \cdot HA_3 = \frac{p_i}{p_s} \cdot UW_2 \end{aligned}$$

Until now we have only considered non-negative absolute differences. What happens if we drop the requirement that *i* is never disfavoured compared with *s* and use the absolute values of the differences in Table 6.1 and Table 6.2? [Hu] (page 107-108) presents the answers. For MF_1 and MF_2 we get the same method, i.e. MF, if we base the pairwise comparisons on the absolute values of the differences. Also the absolute value versions of HM_1 and HM_2 result in the same method, i.e. HM. The reason for these results is the symmetry of the two terms in these 4 pairwise comparisons. The absolute value versions of the pairwise comparisons for SD, HA, EP, and UW are either unworkable or result in MF or HM. 8 are unworkable. It turns out that all pairwise comparisons which can be expressed as MF_1 or HM_1 multiplied by a constant, see (6.28) for the relationships, give MF and HM respectively when the absolute value versions are used. Thus, the absolute value versions of SD_1 and HA_1 imply MF, while the absolute value versions of UW_3 and UW_4 yield HM.

We end this section by constructing what we call weighted pairwise comparisons. Let us multiply MF_1 by the constant $(p_i + p_s)$. The pairwise comparison thus created is:

(6.29)
$$(p_i + p_s)\cdot\left(\frac{a_i}{p_i} - \frac{a_s}{p_s}\right) = \left(a_i - p_i \cdot \frac{a_s}{p_s}\right) + \left(p_s \cdot \frac{a_i}{p_i} - a_s\right)$$

It can be shown by using the same technique as in Example 6.1 that this weighted pairwise comparison implies MF. We recognize the terms in the first parenthesis on the right hand side of (6.29) as SD_1 and the terms in the second parenthesis as HA_1. Recall that SD_1 and HA_1 are the only pairwise comparison tests for SD and HA which have workable absolute value versions. Moreover, both of these lead to MF. Another way of getting the pairwise comparison in (6.29) is by multiplying MF_2 by $(\frac{1}{p_i} + \frac{1}{p_s})$.

We construct a weighted pairwise comparison test which implies HM similarly; either by multiplying HM_1 by $(\frac{1}{p_i} + \frac{1}{p_s})$ or by using the weight $(p_i + p_s)$ on HM_2:

(6.30) $$(p_i + p_s)\cdot(\frac{1}{p_i \cdot a_s} - \frac{1}{p_s \cdot a_i}) = (\frac{1}{a_s} - \frac{p_i}{p_s \cdot a_i}) + (\frac{p_s}{p_i \cdot a_s} - \frac{1}{a_i})$$

The comparison within the first parenthesis on the right hand side is UW_3, while the comparison within the second parenthesis is UW_4. Recall that both UW_3 and UW_4 have absolute value versions which give HM. We have not been able to create weighted pairwise comparisons for other apportionment methods than MF and HM.

Chapter 7: Thresholds and payoffs

The vote share needed to win a seat is an important figure when parties compete for seats. This vote share is known as the threshold. In many countries a party must surpass a threshold fixed by law to win seats in the parliament. Such thresholds are usually in the range 1% - 5%. In this chapter we investigate the thresholds built into different apportionment methods. The threshold for a divisor method is heavily influenced by the size of the first divisor. Since $d_1 = 0$ for SD, HM, and EP, one might say that these methods have a threshold of 0%. The threshold is greater than 0% when MF, HA, or another divisor method with $d_1 > 0$ is used. It is possible to generalize the threshold idea to the vote share needed to win l seats. Then we talk about the payoff for l seats.

The structure of this chapter is as follows: The first section introduces the terminology and gives a brief presentation of the scenarios the payoff functions are derived from. In section 7.2 we deduce payoff functions and threshold formulas for the family of constant parametric divisor methods. We deduce similar formulas for LF in section 7.4. Section 7.3 makes use of a payoff function to analyse the condition called preservation of the majority. The penultimate section presents apportionment methods with built-in thresholds. We try out some of these methods on a real election situation in the last section.

7.1 Terminology

Let us start by issuing a warning: Many of the thresholds found in the literature are incorrect. The correct thresholds and payoff functions for HA, MF, modified MF, and LF are presented in [L&G].

There are two theoretical thresholds for each apportionment method, a lower and an upper. [R] (page 193) calls them **threshold of representation** and **threshold of exclusion** respectively.

Definition 7.1
The **threshold of representation** is the lowest vote share which may give the party a seat.

Definition 7.2
The **threshold of exclusion** is the highest vote share for which a party may fail to win a seat.

Notice that the meaning of the word threshold differs slightly between these two definitions. Normally, the distinction has not any practical consequences. When a party's vote share is below the threshold of representation the party will not be represented, while it is assured of representation if its vote share is above the threshold of exclusion. The interval between these two thresholds, including them both, is the grey zone where the actual threshold will lie. We call this interval the **threshold interval**.

Definition 7.3
Threshold interval = [Threshold of representation , Threshold of exclusion]

A party's vote share can be seen as the price the party has to pay for its seats. The threshold of representation is the lowest price a party might pay and have a chance of getting its first seat, while the threshold of exclusion is the highest price that might be paid without getting a seat. We generalize this pricing idea for the

two theoretical thresholds to situations where a party competes for its *l*th seat. The payoff functions thus created are called **representation payoff** and **exclusion payoff** respectively. These functions are defined for the positive integers $l \in H$. The actual payoff for l seats will lie somewhere in the interval between these two payoff functions. We call this interval, including both end points, **threshold interval for the *l*th seat**. The words "for the *l*th seat" are only used when $l \geq 2$. [L&G] (page 230) define their payoff functions in terms of minimum and maximum payoff for l seats. Our representation payoff for l seats is equal to their minimum payoff for l seats, while our exclusion payoff for l seats corresponds to their maximum payoff for $l - 1$ seats.

Since the description of thresholds and payoff functions is based on a party setting, we need to introduce some notation regarding parties: n is the **number of parties**, N is the set of all parties, and we use the index $j \in N = \{1, 2, ..., n\}$ for the parties. In the constituency context in chapter 1 we assumed that all populations were positive. In this chapter we allow some parties with no votes, i.e. we assume that the party votes (populations) are non-negative $p_j \geq 0$. The total number of votes is denoted p_N. To be possible for a party to win l seats, the house size must be big enough, i.e. $h \geq l \geq 1$. Moreover, we assume that there are at least two parties, i.e. $n \geq 2$. The situation with $h = 1$ and $n = 2$ is trivial. From Definition 7.1 and Definition 7.2 it is clear that the two thresholds are identical and equal to $\frac{1}{2}$, which further means that the threshold interval contains one point, $\frac{1}{2}$. All other situations are more complex than this border case.

The following tie situation is the focus of attention in the deduction of payoff functions for both CP_t and LF: We have a tie which involves all n parties. In all deductions we follow party s which tries to get its *l*th seat. We deduce the function for the representation payoff for l seats from the best case scenario for s: The tie is for the last seat and s wins this seat. The worst case scenario for s determines the function for the exclusion payoff for l seats. In this scenario the tie is for the last $n - 1$ seats and s is the only party which loses out in the tie-break.

7.2 Payoff functions for CP_t

In this section we derive payoff functions for the family of constant parametric divisor methods. We assume that $t > 0$ in these deductions. The following relationships hold for the divisor series given this restriction: $0 < d_1 = t \leq 1$ and $d_l = d_{(l-1)} + 1$ for $l \geq 2$. When setting up the election situations the payoff functions are derived from, we bear in mind Definition 2.2 which sees the divisors as sign posts. With the exception of CP_1 (HA), the distance from the origin to the first sign post is shorter than the distances between neighbouring sign posts, i.e. $d_1 - 0 < d_l - d_{(l-1)}$. CP_1 is the border case because $d_1 = d_l - d_{(l-1)}$. We interpret the distance between the origin and the first sign post as the cost of a party's first seat, the distance between the first and the second signpost as the cost of a party's second seat etc. Thus, when a constant parametric divisor method with $t < 1$ is used, a party's first seat is less expensive than any other seat the party might win. Furthermore, the cost of the lth seat is the same for all $l \geq 2$; with $t = 1$ (HA) this statement holds for $l \geq 1$. Recall that our attention is centered on party s which competes for its lth seat.

The optimal situation for s is that all other parties just fail to win the last seat they compete for. Since the cost is the same for all other seats than the first, we might just as well assume that as many as possible of the other parties compete for their first seat. This is what we do in the deduction of the representation payoff function.

Since the cost of a party's first seat never is larger than the cost of any other seat it might win, it is clear that the worst case scenario for party s is that there are as many other parties as possible which all just succeed in winning their first seat. This explains the assumed sizes of the parties in the deduction of the exclusion payoff function.

Hence, in both scenarios we assume that the bulk of the other parties compete for their first seat. We divide the other parties in two groups as follows: The first

group consists of n - 2 parties which all try to win their first seat, while the second group consists of party j. In both scenarios s and the n - 2 small parties are tied, so the size of each small party can be expressed as $\frac{d_1}{d_l} \cdot p_s$. The size of j, which has the task of picking up the remaining votes and seats, is therefore:

$$p_j = p_N - p_s - (n - 2) \cdot \frac{d_1}{d_l} \cdot p_s = p_N - [d_l + (n - 2) \cdot d_1] \cdot \frac{p_s}{d_l} \tag{7.1}$$

j is also tied with s. To determine the payoff functions we set the quotients for s and j equal to each other. Afterwards we must control that the derived functions do not imply that j has negative support.

We start by deducing the representation payoff formula from the best case scenario for s. Since none of the small parties wins a seat, party j is competing for its seat number $h - (l - 1)$. We set the quotients for s and j equal to each other and get:

$$\frac{p_s}{d_l} = \frac{p_N - p_s - (n-2) \cdot \frac{d_1}{d_l} \cdot p_s}{d_{(h-l+1)}} \Leftrightarrow$$

$$\frac{p_s}{p_N} = \frac{d_l}{d_{(h-l+1)} + d_l + (n-2) \cdot d_1} = \frac{l - 1 + t}{(h-l+1-1+t) + (l-1+t) + (n-2) \cdot t} \Leftrightarrow$$

$$\frac{p_s}{p_N} = \frac{l - 1 + t}{h - 1 + n \cdot t} \tag{7.2}$$

(7.2) is the formula for the representation payoff for a constant parametric divisor method with $t > 0$, i.e. it shows the lowest possible payoff for l seats. To verify that j has not negative support, we look at the number of the divisor in j's quotient. This number cannot be smaller than 1, corresponding to the divisor d_1. Hence, $h - l + 1 \geq 1 \Leftrightarrow l \leq h$, which is satisfied.

By inserting $l = 1$ in (7.2) we get the formula for the threshold of representation:

$$\frac{p_s}{p_N} = \frac{t}{h - 1 + n \cdot t} \tag{7.3}$$

By letting t take on the values $\frac{1}{3}$, $\frac{1}{2}$, and 1 respectively, we find that the threshold of representation is $\frac{1}{3 \cdot h + n - 3}$ for DM, $\frac{1}{2 \cdot h + n - 2}$ for MF, and $\frac{1}{h + n - 1}$ for HA.

We move on to the deduction of the exclusion payoff function. Since all n - 2 small parties get one seat each, j competes for its seat number h - (n - 2) - (l - 1). When s' and j's quotients are set equal to each other we get:

$$\frac{p_s}{d_l} = \frac{p_N - p_s - (n-2)\cdot\frac{d_1}{d_l}\cdot p_s}{d_{(h-n-l+3)}} \Leftrightarrow$$

$$\frac{p_s}{p_N} = \frac{d_l}{d_{(h-n-l+3)} + d_l + (n-2)\cdot d_1} = \frac{l - 1 + t}{(h-n-l+3-1+t) + (l-1+t) + (n-2)\cdot t} \Leftrightarrow$$

$$\frac{p_s}{p_N} = \frac{l - 1 + t}{h + 1 - n\cdot(1 - t)} \tag{7.4}$$

(7.4) is our provisional formula for the exclusion payoff. We check whether j gets at least one seat via the divisor: $h - n - l + 3 \geq 1 \Leftrightarrow n + l - 2 \leq h$. (7.4) is not valid in situations where $n + l - 2 > h$. What happens in such situations is dealt with later.

By substituting 1 for l in (7.4) we get the provisional formula for the threshold of exclusion:

$$\frac{p_s}{p_N} = \frac{t}{h + 1 - n\cdot(1 - t)} \tag{7.5}$$

By letting t take on the values $\frac{1}{3}$, $\frac{1}{2}$, and 1, we get $\frac{1}{3\cdot h - 2\cdot n + 3}$, $\frac{1}{2\cdot h - n + 2}$, and $\frac{1}{h+1}$ as the provisional thresholds of exclusion for DM, MF, and HA respectively.

We notice that the threshold of exclusion for HA is independent of the number of parties competing for seats. The reason is that HA (CP_1) is the only constant parametric divisor method where every seat costs the same. With HA, s' struggle to win its first seat can also be illustrated by the following worst case scenario: j is so big that it threatens to win all h seats, s competes for its first seat, while the other n - 2 parties have zero support. For s to be assured of winning a seat its first quotient must be greater than j's hth quotient:

$$\frac{p_s}{1} > \frac{p_j}{h} = \frac{p_N - p_s}{h} \Leftrightarrow p_s\cdot(h + 1) > p_N \Leftrightarrow \frac{p_s}{p_N} > \frac{1}{h+1}$$

The formulas above have been derived under the assumption that $t > 0$. With SD, i.e. $t = 0$, the first divisor is zero $d_1 = 0$, which gives quotients with zero in the denominator for parties competing for their first seat. It turns out that formulas (7.3) - (7.5) are incorrect for $t = 0$, while (7.2) works well when $l > 1$. By trying out $t = 0$ in (7.3) and (7.5) we get 0% as both the threshold of representation and threshold of exclusion. A threshold of representation equal to zero would have been possible given the artifical and unreasonable assumption that all quotients are equal, i.e. $\frac{p_i}{0} = \frac{0}{0} = \frac{p_i}{1} = \frac{p_i}{2}$ etc. A consequence of such an assumption would have been that a party with zero support would have the same chance of getting the seat in a single member constituency as a party with 100% support. It is correct to say that the threshold of exclusion is 0% for $t = 0$ when $n \leq h$ because a party with positive support, however small, shall get a seat when SD is applied and there are enough seats available. It is unreasonable that all parties with positive support are guaranteed representation when $n \leq h$, so SD should not be used for apportionment of seats among parties.

(7.4) and (7.5) are only valid when $n + l - 2 \leq h$. To complete the picture we study the exclusion payoff for situations where $n \geq h - l + 2$. There is a special worst case scenario for these situations. We start our investigation by studying the threshold of exclusion situation:

s competes for its first seat $l = 1$, so the restriction becomes $n \geq h + 1$. Since there are h seats to compete for, the situation for s cannot be worse than when $h + 1$ parties with identical positive support compete for these seats and s is the only party which loses out in the tie-break. The remaining $n - h - 1$ parties have zero support. If there had been more than $h + 1$ parties with positive support, the maximal vote share needed to win a seat would have decreased. The size of s, as well as every other party with positive support, is $p_s = \frac{p_N}{h+1}$. Hence, $\frac{p_s}{p_N} = \frac{1}{h+1}$ is the threshold of exclusion when $n \geq h + 1$. Since the value of (7.5) is smaller than $\frac{1}{h+1}$ when $n < h + 1$ and $t > 0$, $\frac{1}{h+1}$ is the upper bound for the threshold of exclusion. An alternative way of finding this threshold is by inserting $n = h + 1$ in (7.5).

Thus, every constant parametric divisor method with $t > 0$ has a threshold of exclusion equal to $\frac{1}{h+1}$ when $n \geq h + 1$. This threshold is also valid for $t = 0$ (SD) according to Definition 2.1, because there are not enough seats to give every party one each. We notice that the derived threshold is equal to the general threshold of exclusion for HA and independent of the number of parties competing for seats. The independence of n is natural since parties with zero support cannot influence the threshold. $\frac{1}{h+1}$ as the threshold of exclusion when $n \geq h + 1$ is by no means unique for the constant parametric divisor methods; all proportional apportionment methods have the same threshold. The reason is that all such methods have the same set of possible apportionments for this special worst case scenario.

By combining the threshold of representation from (7.3) with the formulas for the threshold of exclusion, we find the formula for the threshold interval for a constant parametric divisor method with $t > 0$:

(7.6)
$$\left[\frac{t}{h-1+n\cdot t}, \frac{t}{h+1-n\cdot(1-t)}\right] \quad \text{for } n \leq h+1$$
$$\left[\frac{t}{h-1+n\cdot t}, \frac{1}{h+1}\right] \quad \text{for } n \geq h+1$$

We mentioned earlier on that substitution of $h + 1$ for n in (7.5) is a way of finding the upper bound for the threshold of exclusion. Similarly, we substitute $h - l + 2$ for n in (7.4) to determine the upper bound for the exclusion payoff:

(7.7)
$$\frac{p_s}{p_N} = \frac{l-1+t}{h+1-(h-l+2)\cdot(1-t)} = \frac{l-1+t}{l-1+t\cdot(h-l+2)}$$

We observe that $l = 1$, i.e. the threshold of exclusion situation, is the only case where the upper bound for the exclusion payoff is independent of t. The representation payoff function is unchanged from (7.2), so the threshold interval for the lth seat given a constant parametric divisor method with $t > 0$ is:

(7.8)
$$\left[\frac{l-1+t}{h-1+n\cdot t}, \frac{l-1+t}{h+1-n\cdot(1-t)}\right] \quad \text{for } n \leq h-l+2$$
$$\left[\frac{l-1+t}{h-1+n\cdot t}, \frac{l-1+t}{l-1+t\cdot(h-l+2)}\right] \quad \text{for } n \geq h-l+2$$

The payoff functions and thresholds for other divisor methods than the constant parametric ones have to be deduced separately for each method (family of methods). For an example see the penultimate section. Section 7.4 presents the payoff functions for LF.

7.3 Preservation of the majority

The ultimate goal in parliamentary elections is to win the majority of the seats. To keep it simple, let us suppose that the whole country is one constituency. Then it is reasonable to require that a party which gets the majority of the votes should get at least half the seats. This condition is called **preservation of the majority**, [H-A] (page 85):

Definition 7.4
An apportionment method A **preserves the majority** if
$\mathbf{a} \in A(\mathbf{p},h)$ and $\frac{p_j}{p_N} > \frac{1}{2}$ imply $a_j \geq \frac{h}{2}$

Our aim with this section is to determine which constant parametric divisor method(s) that preserve the majority under the most unfavourable circumstances for a party s. We therefore assume that s has got just a little bit more than half the votes. Moreover, the worst case scenario for s is that it has to pay the highest possible price for its seats. The maximum payoff for l seats is the exclusion payoff for $l + 1$ seats. Let $L \in \{0, 1, ..., h\}$. We convert the exclusion payoff function in the right part of (7.8) to a maximum payoff for L seats function by substituting $L + 1$ for l:

(7.9)
$$\begin{cases} \dfrac{L+t}{h+1-n\cdot(1-t)} & \text{for } L \leq h-n+1 \\ \dfrac{L+t}{L+t\cdot(h-L+1)} & \text{for } L \geq h-n+1 \end{cases}$$

By letting the functions in (7.9) be equal to $\frac{1}{2}$, we deduce the formulas for L in situations where a party has got half the votes:

(7.10) $L = \frac{h - n + 1}{2} + t \cdot (\frac{n}{2} - 1)$ for $L \leq h - n + 1$

(7.11) $L = \frac{t \cdot (h - 1)}{1 + t}$ for $L \geq h - n + 1$

L is the lowest number of seats a party with half the votes may get. The expressions on the right hand sides of (7.10) and (7.11) result in rational, integer, or real numbers. However, L is an integer, so the actual right hand side value must be rounded up to the nearest integer. Thus, if the actual right hand side value is $a \in \Re$, the party will get at least $\lceil a \rceil$ seats. As stated above, we assume that party s has got just a little bit more than half the votes. To get formulas suitable for s, we increase the right hand sides of (7.10) and (7.11) by an infinitely small number $\varepsilon > 0$:

(7.12) $L_s = \frac{h - n + 1}{2} + t \cdot (\frac{n}{2} - 1) + \varepsilon$ for $L_s \leq h - n + 1$

(7.13) $L_s = \frac{t \cdot (h - 1)}{1 + t} + \varepsilon$ for $L_s \geq h - n + 1$

In our analysis of (7.12) and (7.13) we distinguish between even and odd numbered house sizes. Moreover, we utilize that L_s is an integer. When h is an even number, $\frac{h}{2}$ is integer and the premise for preservation of the majority $a_s \geq \frac{h}{2}$ translates into $L_s > \frac{h}{2} - 1$. On the other hand, when h is an odd number, $\frac{h}{2}$ is some integer plus a half. Then the premise can be transformed to $L_s > \frac{h}{2} - \frac{1}{2}$. The two inequalities regarding L_s are equivalent to $L > \frac{h}{2} - 1 - \varepsilon$ and $L > \frac{h}{2} - \frac{1}{2} - \varepsilon$ respectively. To get inequalities which are easier to work with than these, we remove ε from the right hand sides. We make up for this removal by substituting $\geq$ for $>$ in both inequalities. This leaves us with the following conditions for preservation of the majority:

(7.14) $L \geq \frac{h}{2} - 1$ for h an even number

(7.15) $L \geq \frac{h}{2} - \frac{1}{2}$ for h an odd number

As seen from (7.10) - (7.11), there are different formulas for L dependent on the size of L. Instead of expressing the dividing line between these formulas in terms of L, i.e. as $L = h - n + 1$, we utilize the expressions for L in (7.14) and (7.15) to eliminate L from the dividing line formula. In the even number case we substitute equality for inequality in (7.14) and equate this expression for L with the existing formula for the dividing line:

$$(7.16) \qquad \frac{h}{2} - 1 = h - n + 1 \Leftrightarrow n = \frac{h}{2} + 2$$

(7.16) is the dividing line between the formulas when h is an even number. Similarly, we find $n = \frac{h}{2} + \frac{3}{2}$ to be the dividing line in the odd number case. These dividing lines are of interest below.

Then we are ready to investigate the different cases regarding preservation of the majority. We utilize formulas (7.10) - (7.11) and conditions (7.14) - (7.15) in this investigation. In turn we put the expression for L from one of the formulas into one of the conditions. Each resulting inequality is thereafter solved with respect to $0 < t \leq 1$. We start with formula (7.10). In the even number case we get:

$$\frac{h - n + 1}{2} + t \cdot (\frac{n}{2} - 1) \geq \frac{h}{2} - 1 \Leftrightarrow$$

$$(7.17) \qquad t \geq \frac{n - 3}{n - 2} \qquad \text{for } 2 < n \leq \frac{h}{2} + 2 \text{ and } h \text{ even}$$

When h is an odd number, the condition becomes:

$$\frac{h - n + 1}{2} + t \cdot (\frac{n}{2} - 1) \geq \frac{h - 1}{2} \Leftrightarrow$$

$$(7.18) \qquad t = 1 \qquad \text{for } 2 < n \leq \frac{h + 3}{2} \text{ and } h \text{ odd}$$

We now turn our attention to formula (7.11). For even numbered house sizes the condition is:

$$\frac{t\cdot(h-1)}{1+t} \geq \frac{h}{2} - 1 \Leftrightarrow$$

(7.19) $$t \geq \frac{h-2}{h} \qquad \text{for } n \geq \frac{h}{2} + 2 > 2 \text{ and } h \text{ even}$$

The odd number case results in:

$$\frac{t\cdot(h-1)}{1+t} \geq \frac{h-1}{2} \Leftrightarrow$$

(7.20) $$t = 1 \qquad \text{for } n \geq \frac{h+3}{2} > 2 \text{ and } h \text{ odd}$$

It is time to summarize our findings. We start with some trivial election situations which are not covered by (7.17) - (7.20). Every constant parametric divisor method, in fact every proportional apportionment method, preserves the majority when $n = 2$ and/or $h \leq 2$. We continue with the case that h is an even number larger than 2. Constant parametric divisor methods with $\frac{n-3}{n-2} \leq t \leq 1$ preserve the majority for election situations where $2 < n \leq \frac{h}{2} + 2$. Moreover, the majority is also preserved when $\frac{h-2}{h} \leq t \leq 1$ for election situations where $n \geq \frac{h}{2} + 2 > 2$. Finally, we consider the case that h is an odd number larger than 1. Only the constant parametric divisor method with $t = 1$, i.e. HA, preserves the majority in these situations. Thus, CP_1 (HA) is the only constant parametric divisor method which preserves the majority independently of the house size and number of parties. Example 7.1 illustrates how other constant parametric divisor methods fail to satisfy the condition:

Example 7.1

Consider the quota vector $\mathbf{q} = (\frac{751}{1500}, \frac{749}{3000}, \frac{749}{3000})$. The first party has got more than half the votes, so it must get at least 2 seats for the majority to be preserved. Let us apply the constant parametric divisor method with $t = \frac{99}{100}$, i.e. almost 1, for this election situation. The resultant apportionment is $\mathbf{a} = (1, 1, 1)$, so this method does not preserve the majority.

We round off this section by presenting results for the apportionment method dealt with in the next section: LF does preserve the majority for even numbered house sizes. However, it does not for odd numbered house sizes where $h \geq 3$; just consider the election situation with $\mathbf{q} = (\frac{8}{5}, \frac{7}{10}, \frac{7}{10})$.

7.4 Payoff functions for LF

We now switch our attention to LF. For this method the assumption that all parties are tied means that they have equal fractions. In the deduction of payoff functions for LF we only have to consider party s explicitly.

The lowest payoff which may result in l seats is determined from the best case scenario: There is only one seat for distribution to the fractions, so the fractional part of the quota is $\frac{1}{n}$ for every party. s wins this seat in the tie-break. In addition, s wins l - 1 seats for the integer part of its quota.

$$q_s = \frac{p_s \cdot h}{p_N} = l - 1 + \frac{1}{n} \Leftrightarrow$$

$$\frac{p_s}{p_N} = \frac{l - 1}{h} + \frac{1}{n \cdot h} \tag{7.21}$$

This is the representation payoff function. The first term on the right hand side of (7.21) is the proportional vote share for l - 1 seats, while the second term is the extra vote share needed to be able to take the only seat distributed to the fractional parts. $\frac{1}{n \cdot h}$ is therefore the threshold of representation for LF.

The highest payoff which may fail to win l seats is deduced from the worst case scenario: There are as many seats as possible left after step 1 of Algorithm 1.1. s loses out in the tie-break such that these n - 1 seats go to the other parties. All parties are tied, so their fractions are $\frac{n - 1}{n}$. Thus, the provisional exclusion payoff for LF is:

(7.22) $$\frac{p_s}{p_N} = \frac{l - 1}{h} + \frac{n - 1}{n \cdot h}$$

The second term on the right hand side of (7.22) is the provisional treshold of exclusion for LF. To be able to distribute n - 1 seats to the fractions in addition to l - 1 seats to s, the house size must be at least (l - 1) + (n - 1). Thus, (7.22) is only valid when $h \geq l + n - 2$. The upper bound for the exclusion payoff is found by substituting $h - l + 2$ for n in (7.22):

(7.23) $$\frac{p_s}{p_N} = \frac{l - 1}{h} + \frac{h - (l - 1)}{h \cdot (h - l + 2)}$$

By inserting $l = 1$ in (7.23) we again encounter the upper bound $\frac{1}{h+1}$ for the threshold of exclusion for proportional apportionment methods. We end this examination of payoff functions for LF by stating the formula for the threshold interval for the lth seat:

(7.24) $$[\frac{l - 1}{h} + \frac{1}{n \cdot h} , \min (\frac{l - 1}{h} + \frac{n - 1}{n \cdot h} , \frac{l - 1}{h} + \frac{h - (l - 1)}{h \cdot (h - l + 2)})]$$

7.5 Threshold methods

At the beginning of the chapter we mentioned that many countries have law determined thresholds in the range 1% - 5%. In this section we try to construct divisor series with similar thresholds. The starting point for all constructions is a strict divisor series which satisfies Definition 2.2. We denote the divisors in the original series d, while the divisors in the constructed divisor series are denoted D. f is a positive integer which denotes the number of divisors with special status. Be aware that the constructed divisor series only satisfy Definition 2.1, not Definition 2.2.

Let us take a strict divisor series and eliminate its first f divisors. We call the apportionment method defined by the resulting divisor series a **cutted divisor method.**

Definition 7.5
We define an apportionment method as a **cutted divisor method** if its lth divisor is defined as the $(f + l)$-th divisor of a divisor method which satisfies Definition 2.2:

(7.25) $$D_l = d_{(f+l)}$$

With HA as the original method and $f = 5$, the divisors in the corresponding cutted divisor series are $D_1 = d_6 = 6$, $D_2 = 7$ etc. A cutted divisor method with $D_1 > 1$ is not exact. Moreover, it overrepresents large parties. There is only one instance where a cutted divisor method is exact. It occurs for the cutted divisor method based on SD and with $f = 1$, because then the resultant apportionment method is HA.

A better idea than the cutted divisor method is to let all parties which surpass the threshold get their proportional share of the seats. This is achieved by letting the first f divisors be equal to the $(f + 1)$-th divisor of the original method, while the divisors from number $f + 1$ and upwards are set equal to the corresponding divisor in the original strict divisor series. We call this kind of apportionment method a **threshold divisor method.**

Definition 7.6
An apportionment method is a **threshold divisor method** if its divisor series is equal to the divisor series of a method which satisfies Definition 2.2, except for the first f divisors which all are equal to the $(f + 1)$-th divisor of the original divisor series:

(7.26) $$D_1 = \ldots = D_f = d_{(f+1)}$$
$$D_l = d_l \qquad \text{for } l > f$$

With $f = 6$ and $d_l = l - \frac{1}{2}$, i.e. MF, the first seven divisors in the corresponding threshold divisor series are all equal to $6\frac{1}{2}$, while $D_8 = 7\frac{1}{2}$ etc. A threshold divisor method is proportional for the parties which surpass the threshold. The main difference between a cutted divisor method and a threshold divisor method is that a party which just surpasses the threshold, i.e. D_1, wins up to $f + 1$ seats with a threshold divisor method, but only one seat with a cutted divisor method.

Below we deduce formulas for the threshold of representation and threshold of exclusion for the two types of apportionment methods defined above. The underlying assumption in these deductions is that the original strict divisor series stems from a constant parametric divisor method.

The best case scenario for s with a cutted divisor method is the same as with a pure constant parametric divisor method. We therefore find the threshold of representation by inserting $l = 1$ into the formula leading up to equation (7.2) and converting the divisors:

$$\frac{p_s}{p_N} = \frac{D_1}{(n-1)\cdot D_1 + D_h} = \frac{d_{(f+1)}}{(n-1)\cdot d_{(f+1)} + d_{(f+h)}} = \frac{f+t}{(n-1)\cdot(f+t) + (f+h-1+t)} \Leftrightarrow$$

$$\frac{p_s}{p_N} = \frac{f+t}{h-1+n\cdot(f+t)} \tag{7.27}$$

We must verify that j does not have negative support. Since D_h is the divisor used in j's quotient, (7.27) is valid when $h \geq 1$, i.e. always.

With a cutted divisor method based on a constant parametric divisor method, the cost of a party's first seat is never lower than the cost of any other seat. The worst case scenario for s is therefore a large party j and zero support for the remaining parties. From this scenario we deduce the threshold of exclusion for a cutted divisor method:

$$\frac{p_s}{D_1} = \frac{p_j}{D_h} = \frac{p_N - p_s}{D_h} \Leftrightarrow \frac{p_s}{p_N} = \frac{D_1}{D_1 + D_h} = \frac{f+t}{(f+t) + (f+h-1+t)} \Leftrightarrow$$

$$\frac{p_s}{p_N} = \frac{f+t}{h-1+2\cdot(f+t)} \tag{7.28}$$

We observe that (7.28) is independent of the number of parties. This is natural since only two parties were needed to create the worst case scenario.

We move on to the threshold of representation for a threshold divisor method. This threshold is determined from the same best case scenario as for a cutted divisor method. However, we must take into consideration that D_h is defined differently with a threshold divisor method:

$$\frac{p_s}{p_N} = \frac{D_1}{(n-1)\cdot D_1 + D_h} = \frac{d_{(f+1)}}{(n-1)\cdot d_{(f+1)} + d_h} = \frac{f+t}{(n-1)\cdot(f+t) + (h-1+t)} \Leftrightarrow$$

$$\frac{p_s}{p_N} = \frac{f+t}{h-(f+1)+n\cdot(f+t)} \tag{7.29}$$

The divisors in the quotients $\frac{p_s}{D_1} = \frac{p_s}{d_{(f+1)}}$ and $\frac{p_j}{D_h} = \frac{p_j}{d_h}$ tell us that only the largest party surpasses the threshold and wins seats when $f+1 > h$, i.e. when $f \geq h$.

Next we deduce the threshold of exclusion for a threshold divisor method. The use of divisors from a constant parametric divisor series means that the average cost of a party's first $f+1$ seats never is higher than the cost of each of the other seats the party might win. The worst case scenario for s is therefore the same as when a pure constant parametric divisor method is used. What is different is that the other small parties now get $f+1$ seats each.

$$\frac{p_s}{D_1} = \frac{p_N - (n-1)\cdot p_s}{D_{[h-(n-2)\cdot(f+1)]}} \Leftrightarrow \frac{p_s}{p_N} = \frac{f+t}{(n-1)\cdot(f+t) + (h-(n-2)\cdot(f+1)-1+t)} \Leftrightarrow$$

$$\frac{p_s}{p_N} = \frac{f+t}{h+f+1-n\cdot(1-t)} \tag{7.30}$$

For party j to win at least $f+1$ seats it cannot be smaller than s. We check whether this condition is satisfied via the divisor numbers for j and s:

$$h-(n-2)\cdot(f+1) \geq f+1 \Leftrightarrow h \geq (n-1)\cdot(f+1)$$

This inequality must be satisfied for (7.30) to be valid. When $h < (n - 1)\cdot(f + 1)$ there are not enough seats to let j and the other small parties get $f + 1$ each. The worst case scenario in such situations is that an appropriate number of small parties have zero support. This scenario determines the upper bound for the threshold of exclusion for a threshold divisor method based on a constant parametric divisor method. We find this upper bound by inserting $n = \frac{h+f+1}{f+1}$ in (7.30) which yields:

(7.31) $$\frac{f+1}{h+f+1}$$

In the deduction of (7.30) and (7.31) we implicitly made the assumption that a small party which wins a tie-break gets $f + 1$ seats. A more realistic assumption regarding small parties tied "at the threshold" of a threshold divisor method is that seats are split as evenly as possible between these parties. So if there are 6 identical parties, $h = 20$, and $f = 3$, one should give each party 3 or 4 seats rather than give five of the parties 4 seats each and leave the last party without representation.

7.6 Possible threshold methods for Norway

In Norway a party must have at least 4% support to be eligible for the apportionment of supplementary seats. Moreover, there are 165 seats in the Storting and the supplementary seats are apportioned with modified MF. We want to find both a cutted divisor method and a threshold divisor method with thresholds of approximately 4%. The divisor series for MF ($CP_{0,5}$) is used as the basis for these methods. Our view is that a party should always be guaranteed representation if it reaches the threshold fixed by law. We therefore use the formulas for the threshold of exclusion in the determination process.

The f value for the cutted divisor method based on MF is determined by putting the relevant data into (7.28) and solving for f:

$$\frac{f+\frac{1}{2}}{165 - 1 + 2\cdot(f+\frac{1}{2})} \leq \frac{4}{100} \Leftrightarrow f \leq \frac{305}{46} \approx 6{,}63$$

Since f has to be integer, $f = 6$.

To determine the appropriate f value for the threshold divisor method based on MF we have to make an assumption regarding the number of parties. $n = 7$ is chosen because this was the number of parties with a realistic chance of surpassing the threshold in 1993. Utilization of (7.30) yields:

$$\frac{f+\frac{1}{2}}{165 + f + 1 - 7\cdot(1 - \frac{1}{2})} \leq \frac{1}{25} \Leftrightarrow f \leq \frac{25}{4} = 6{,}25$$

$f = 6$ is the highest integer which satisfies this inequality. To see how the number of parties influences the inequality we let $n = 10$, which is nearer to the truth. The result is a lowering of the right hand side to about 6,19, so the number of parties does not influence f very much. We do not violate the condition for validity of (7.30) with the calculations above, because $165 \geq (n - 1)\cdot(f + 1)$ for the n and f values considered.

There were 6 parties which surpassed the threshold of 4% in the general election in 1993 and one party, V (Liberals), which got 3,6% of the votes. We now do the necessary calculations to check whether V would have won seats with the suggested methods. The total number of votes each party got in the election are given in Table 7.1:

Table 7.1

Parties	A	FrP	H	KrF	SP	SV	V	Others
Votes	908724	154497	419373	193885	412187	194633	88985	89665
%	36,91%	6,28%	17,03%	7,88%	16,74%	7,91%	3,61%	3,64%
$f = 6$	66,38	11,29	30,63	14,16	30,11	14,22	6,5	
$f = 5$	56,17	9,55	25,92	11,98	25,48	12,03	5,5	

Vote totals in the Norwegian election in 1993 and party quotients with the common divisors $z = 13690$ ($f = 6$) and $z = 16179\frac{1}{11}$ ($f = 5$).

The first quotient for V is $\frac{pv}{D_1} = \frac{88985}{\frac{13}{2}} = 13690$ with both threshold methods based on $f = 6$. For V to win seat(s), this quotient has to be among the 165 highest quotients. We divide all vote totals by 13690 and find the values in the row marked $f = 6$. These values are nothing else than what we called constituency quotients in section 2.7. It is natural to use the name party quotients here. Moreover, we recognize 13690 as a common divisor z.

We start with the threshold divisor method based on MF. To determine the number of quotients higher than 13690, we round the party quotients to the nearest integer. The other parties have $66 + 11 + 31 + 14 + 30 + 14 = 166$ quotients which are better than V's first quotient, so V would not have been represented. Recall that the determination of $f = 6$ earlier in the section was based on the assumption that a party with 4% support should always be represented. This and the fact that f has to be integer are the reasons why V almost gets represented although its support is below 4%.

With the cutted divisor method the story is different. Since $f = 6$ here means that the first 6 divisors in MF have been eliminated, there are only $166 - (6 \cdot 6) = 130$ quotients which are higher than 13690. Thus, V would have won seats if this cutted divisor method had been used. Moreover, this means that the actual threshold is lower than 3,6% in this case.

Let us decrease f for the threshold divisor method to 5 and see what happens. The highest quotient for V then becomes $\frac{88985}{\frac{11}{2}} = 16179\frac{1}{11}$. From the party quotients in the row marked $f = 5$ we find that there are 141 quotients which are better. Thus, V would have won seats if the threshold divisor method based on MF and with $f = 5$ had been used.

Chapter 8: Choice of apportionment method

It is time to raise the question: Which apportionment method should be used? The answer depends on what kind of apportionment problem we are facing and our goal(s) with the apportionment. In this chapter we will see that there are other considerations connected to the apportionment of seats among constituencies than to the apportionment of seats among parties. We call the first kind of apportionment **constituency apportionment** and the second **party apportionment**. They are treated in sections 8.1 and 8.3 respectively. In section 8.2 we describe the current situation in Norway regarding constituency apportionment and propose some improvements. Section 8.4 illustrates how it is possible to take advantage of the current Norwegian election law. In section 8.5 we look at house sizes in the Nordic countries before we in the last section explain the motivation for the matrix apportionment problem, which is part II of this dissertation.

8.1 Constituency apportionment

It is highly unreasonable if a constituency does not get any representative in a general election. The voters in such a constituency would have been deprived of their voting rights. Thus, the house size should not under any circumstances be lower than the number of constituencies, i.e. $h \geq m$. To make sure that every constituency is represented, one can introduce a minimum representation. An example is the US where every state is guaranteed one seat in the House of Representatives. A slightly different way of ensuring every constituency representation is to use a divisor method with the first divisor equal to zero, like SD, HM, or EP. Application of a divisor method with $d_1 = 0$ and $d_2 > 0$ is an indirect way of introducing a minimum representation of one seat. Notice that a

minimum representation of one seat will favour the very small constituencies.

It is quite common to favour small constituencies in the constituency apportionment. An example is the Icelandic constituency apportionment, which is fixed by law. This apportionment greatly disfavours the two most populous constituencies, Reykjavík and Reykjanes. They have only 49% of the seats in the Althing, although they accounted for 65% of the votes cast in the general election in 1995.

Many countries want to favour rural constituencies. A rural constituency is often synonymous with a constituency with a small population. If this is the case for every rural constituency, some apportionment method which favours small constituencies will create the desired favouring. SD is a method which comes to mind. Sometimes SD does not favour the small constituencies as much as one wishes. A possibility then is the family of apportionment methods we call **extended divisor methods**. The divisor series for an extended divisor method consists of one or more divisors equal to 0 plus the divisors from a strict divisor series which satisfies Definition 2.2. Regarding notation, we let D, d, and f stand for the same as in section 7.5. Moreover, we permit division by zero the same way as in Definition 2.1

Definition 8.1
We define an apportionment method as an **extended divisor method** if its first f divisors all are equal to zero, while its $(f + l)$-th divisor is defined as the lth divisor of a divisor method which satisfies Definition 2.2:

$$(8.1) \qquad \begin{aligned} D_1 &= \ldots = D_f = 0 \\ D_{(f+l)} &= d_l \end{aligned}$$

Since the first f divisors of an extended divisor method are equal to zero, there should be at least $m \cdot f$ seats for distribution when such a method is used. The apportionment with an extended divisor method can be determined as follows: Start by giving each constituency f seats. There are $h - m \cdot f$ seats left after this distribution. Apportion these with the original divisor method with divisors d_l.

An extended divisor method is not exact. The only exception to this rule is the extended divisor method based on the divisor series for HA and with $f = 1$, which is nothing else than SD.

A way of favouring special types of constituencies is to base the apportionment on a constructed measure like they do in Denmark. The Danish measure is made up of three parts; population, number of voters, and land area. Other descriptive data which might be used to form a constructed measure are distance to the parliament and inhabitants per square kilometer. We propose a constructed measure for Norway in the next section. The weighting of the different parts of a constructed measure may have a considerable effect on the apportionment. This opens for manipulation of the apportionment through the weight values. It may therefore be difficult to reach an agreement about these values. Since the use of a constructed measure brings the most important reasons for favouring special types of constituencies into the basis for the apportionment, it is natural to carry out the apportionment with an unbiased apportionment method. Of the two candidates MF and LF, MF is preferable because of the paradoxes connected to LF. The fact that MF does not always stay within the quota is only a minor disadvantage with this method.

Some countries fix the representation of each constituency by law. This is an undesirable solution. The experience with such a scheme in Norway and Iceland is that changes in the population distribution influence the constituency apportionment very slowly or not at all. A much better solution is to fix by law which apportionment method to use and which basis (constructed measure) to apply it on. The advantage of such a scheme is that fluctuations in the relevant basis (bases) are reflected in the constituency apportionment immediately.

8.2 Constituency apportionment in Norway

With the exception of 8 seats, the Norwegian constituency apportionment is fixed by law. Today's fixed representation scheme of the constituencies relates back to the early 1950s. There were 150 seats in the Storting in 1953, all fixed. Since then the fixed representation of the two most central constituencies, Oslo and Akershus, has been increased in two steps until a house size of 157 was reached. The last change in the house size occured prior to the general election in 1989, when 8 supplementary seats were introduced. Thus, at the moment the Storting has 165 seats. The supplementary seats are special because they are not preassigned to any constituency. Their placement among the constituencies depends on several aspects of the election result.

In the rest of this section we give examples of how the apportionment methods described in section 8.1 would have distributed the 165 seats in the Norwegian Storting. The main basis for these constituency apportionments is the number of eligible voters in 1993. In Table 8.1 below we present apportionments made by some extended divisor methods. Later on we turn our attention to apportionments based on a constructed measure.

Comments on Table 8.1: The column with the heading "Actual Seats" contains the number of seats each constituency ended up with after the general election in 1993. In the four rightmost columns we present the apportionments one would have got with extended divisor methods based on HA. We have included the HA apportionment ($f = 0$) in the table although HA is not an extended divisor method. As f increases, the small constituencies are favoured more and more. It is also interesting to notice that the largest discrepancy between the apportionments with the extended divisor methods and the actual apportionment occurs for Nordland.

Table 8.1

Constituency	Eligible Voters	Actual Seats	(HA) $f = 0$	(SD) $f = 1$	$f = 2$	$f = 3$
Oslo	359085	17	19	18	16	15
Akershus	319318	14	17	16	15	14
Hordaland	312033	15	16	15	14	14
Rogaland	249445	11	13	12	12	11
Sør-Trøndelag	194755	10	10	10	10	10
Østfold	185495	8	9	9	9	9
Nordland	184446	12	9	9	9	9
Møre og Romsdal	180856	10	9	9	9	9
Buskerud	173890	8	9	9	9	9
Vestfold	154644	7	8	8	8	8
Hedmark	147211	8	7	8	8	8
Oppland	143342	7	7	7	7	8
Telemark	125923	8	6	7	7	7
Troms	114007	6	6	6	6	7
Vest-Agder	107681	5	5	6	6	6
Nord-Trøndelag	96728	6	5	5	6	6
Sogn og Fjordane	80357	5	4	4	5	5
Aust-Agder	73439	4	3	4	5	5
Finnmark	57302	4	3	3	4	5
SUM	3259957	165				

Constituency apportionments with HA and extended divisor methods based on HA.

The time has come to show how constructed measures can be used as bases for apportionments. As mentioned in chapter 1, Denmark employs the constructed measure: Population + Eligible voters + 20·Area. The constructed measure we find most interesting for Norway consists of three parts; the number of eligible voters, the (road) distance to the capital Oslo, and the percentage of people living in sparsely populated areas, later referred to as rural percentage. Let us briefly describe our reasons for recommending this constructed measure: The basic part of every constructed measure intended for constituency apportionment is some measure of population. Our view is that only the people entitled to vote should be taken into account, which explains our use of the number of eligible voters. However, the use of population instead does not normally alter the apportionment much because constituency populations and number of eligible voters tend to be approximately proportional. Next we comment on the other parts of a constructed measure. These parts serve as correctives to the basic part by favouring special types of constituencies based on objective measures. We see it as important that the number of such parts is kept down. The idea behind the inclusion of

distance to Oslo as a part is to compensate the constituencies in the northern part of Norway for their geographical "backwardness" in relation to the capital. We have included the rural percentage to favour rural constituencies. Ideally, we should have used the percentage of voters living in sparsely populated areas. An alternative measure of rurality is land area, but we see the rural percentage as more suitable for Norway. One reason for this is that the northern constituencies which already are most favoured by the distance part also have among the largest areas. It is undesirable with a high degree of correlation between different parts. We end this paragraph with an interpretation of the two "favouring" parts: The distance part takes care of the "rurality" of a constituency compared to the capital, while the rural percentage expresses the rurality within the constituency.

To illustrate the proposed constructed measure, we need data. Here follows a description of the utilized data: We use the number of eligible voters in 1993, but round these numbers to the nearest thousand. The road distance from the administrative town of a constituency to Oslo is the basis for the distance part. Be aware that we subjectively round this distance to one of the nearest multiples of hundred kilometers. The subjective rounding takes into consideration whether the administrative town is situated farther from or nearer to Oslo than the geographical centre of the constituency. Regarding the rural percentage, it is worth mentioning that the Norwegian definition of a densely populated area is an area with a population of at least 200 and with no more than 50 metres between the houses. Sparsely populated areas are the complement of densely populated areas. We base the rural percentage on the 1990 census and give it with one decimal accuracy. The utilized data for the three parts of the proposed constructed measure are presented in the left part of Table 8.2 below. From these data we calculate the value of the constructed measure for each constituency as follows:

$$(8.2) \qquad w_V \cdot \text{Voters} + w_d \cdot \text{Distance} + w_r \cdot \text{Rural percentage}$$

where w_V, w_d, and w_r are positive weights. We use MF to apportion seats based on different weight combinations. Because MF is homogeneous (scale indepen-

dent), we have one degree of freedom in the choice of weights. We set $w_v = 1$ and vary the values of the two other weights. NB. The column with $w_d = w_r = 0$ in Table 8.2 shows the MF apportionment based on the unrounded numbers of eligible voters, i.e. the numbers found in Table 8.1. All other MF apportionments in Table 8.2 are based on constructed measures found by multiplying the weights given in the uppermost part of the table with the rounded values in the "Votes", "Distance", and "Rurality" columns.

Table 8.2

Constituency	Actual apportionment	Voters (in 1000)	Distance (in 100)	Rurality (in %)	Wd Wr	0 0	0,5 0,1	0,5 0,2	1 0,1	1 0,2
Oslo	17	359	0	0,6		18	18	17	17	17
Akershus	14	319	0	15,1		16	16	15	16	15
Hordaland	15	312	5	28,5		16	16	15	16	15
Rogaland	11	249	5	27,6		13	12	12	12	12
Sør-Trøndelag	10	195	5	27,5		10	10	10	10	10
Østfold	8	185	1	20,9		9	9	9	9	9
Nordland	12	184	13	37,3		9	10	10	10	10
Møre og Romsdal	10	181	5	41,7		9	9	9	9	9
Buskerud	8	174	1	25,5		9	9	9	9	9
Vestfold	7	155	1	21,0		8	8	8	8	8
Hedmark	8	147	1	49,4		7	7	8	7	7
Oppland	7	143	2	51,1		7	7	7	7	7
Telemark	8	126	2	28,0		6	6	6	6	6
Troms	6	114	17	40,1		6	6	6	7	7
Vest-Agder	5	108	3	24,2		6	6	6	5	6
Nord-Trøndelag	6	97	7	49,0		5	5	5	5	5
Sogn og Fjordane	5	80	5	55,2		4	4	5	4	5
Aust-Agder	4	73	3	39,0		4	4	4	4	4
Finnmark	4	57	22	27,8		3	3	4	4	4

MF apportionments based on (8.2), where w_v always is set equal to 1.

Let us first comment on the MF apportionment based only on eligible voters, i.e. the apportionment in the column with $w_d = w_r = 0$. This is presumably the most unbiased apportionment achievable for Norwy in 1993, so it is interesting to notice that the actual apportionment deviates quite much. In the following we compare the constructed measure apportionments with the "pure" MF approtionment. We observe that the constituencies in the northern part of Norway, i.e. Finnmark, Troms, and Nordland, are the main beneficiaries of the constructed measures, while the most populous constituencies are the losers. As intended, constituencies far from the capital or with a large percentage of people living in sparsely populated areas have gained. However, none of our constructed measures based on (8.2) are fully able to explain the actual apportionment in 1993. This is

not surprising; [S] (page 33→) shows that the apportionment prescribed by law in 1953 cannot be explained by a MF apportionment based on a constructed measure consisting of voters, distance, and land area.

The last two parts in (8.2) are only related to the constituency, not to the individual voters in the constituency. An alternative approach is to scale these parts by the number of voters in the constituency. With this approach (8.2) would look like this:

(8.3) $$[w_v + \overline{w}_d \cdot \text{Distance} + \overline{w}_r \cdot \text{Rural percentage}] \cdot \text{Voters}$$

where w_v can be set equal to 1 and $\overline{w}_d$ and $\overline{w}_r$ are new weights different from w_d and w_r. An example: Let $w_v = 1$, $\overline{w}_d = 0{,}02$, and $\overline{w}_r = 0{,}15$. With these weights a voter in a sparsely populated area of Hordaland is $\frac{1 + 0{,}02 \cdot 5 + 0{,}15 \cdot 1}{1 + 0{,}02 \cdot 0 + 0{,}15 \cdot 0} - 1 = 25\%$ more important than a voter in central Oslo. (8.3) entails a kind of classification of individual voters and may therefore be seen as unsuitable. The approach in (8.2) is presumably least controversial politically.

8.3 Party apportionment

A guaranteed minimum representation for parties is not reasonable. It could rather be a question of favouring large parties to facilitate the formation of stable governments. Constant parametric divisor methods with $t > 0{,}5$, including HA (CP_1), are apportionment methods which favour large parties. Another way of favouring large parties is to disfavour small parties. Introduction of thresholds at the national and/or constituency level is one way of doing this. Most countries with a proportional representation system have a threshold fixed by law, but the magnitude of this threshold varies. The threshold at the national level is 4% in Norway, but only 2% in Denmark, usually resulting in more small parties in the Danish Folketing than in the Norwegian Storting. From chapter 7 we recall that each apportionment method has a built-in threshold. The use of modified MF in

Norway results in a higher threshold at the constituency level than the use of MF would have done.

In countries with many small sized constituencies even an approximately unbiased apportionment method like LF or MF will normally favour the large parties on a nationwide scale. The extreme regarding small sized constituencies is one seat constituencies, i.e. plurality vote, as in the United Kingdom. In the UK it is not unusual that the largest party wins a considerable majority in the House of Commons with less than 50% of the votes. Election results in different constituencies are normally strongly correlated. Therefore, a slight favouring of large parties in many constituencies accumulates to a greater favouring at the national level.

Let us illustrate why the constituency sizes in Norway favour large parties. In this illustration we assume that MF is used for the apportionment to the parties. 15 of the 19 Norwegian constituencies have between 4 and 10 seats for distribution. With the help of formula (7.6) from the previous chapter we find that the threshold interval for a constituency size of 4 seats is $[\frac{1}{10}, \frac{1}{6}]$ with 4 parties and $[\frac{1}{16}, \frac{1}{5}]$ with 10 parties. The corresponding threshold intervals when there are 10 seats for distribution are $[\frac{1}{22}, \frac{1}{18}]$ and $[\frac{1}{28}, \frac{1}{12}]$ respectively. From the thresholds of representation we see that parties with less than 3,5% support have not even a theoretical chance of winning a seat. The fact that modified MF is the apportionment method used in Norway makes it even harder for a small party to get represented. Moreover, the actual threshold can be quite much higher than the threshold of representation, which does not help small parties either. This analysis demonstrates the disfavouring of small parties within each of the 15 constituencies with a size of 10 seats or less. Added together this amounts to a large disfavouring of small parties. Let us exemplify the disfavouring: Consider a party with a vote share of 4% in each of the 15 constituencies. There are a total of 103 seats for distribution in these constituencies, so the party should get about 4 seats for the apportionment to be proportional. However, most likely it will not be apportioned any seat.

8.4 Exploiting the Norwegian election system

The introduction of 8 supplementary seats has made the Norwegian election system fairer than it was, but there is still a long way to go. This section deals with the possibility of exploiting the rules for the apportionment of supplementary seats. The competition for these seats takes place at the national level, where they are apportioned to the parties with the highest national quotients. When these national quotients are calculated, the seats won by the parties within the constituencies are taken into consideration. We call seats which are preassigned to a constituency fixed seats. As the illustrations in the previous section gave an indication of, the distribution of fixed seats among the constituencies is disproportional. A consequence of this disproportionality is that the average number of voters (people) behind each seat differs a great deal from one constituency to another. Furthermore, this means that a party may gain in the apportionment of supplementary seats by operating as two separate "parties", see [G] (page 244). The second "party" runs in constituencies which have much more voters behind each seat than the national average, while the first "party" runs in the remaining constituencies. Below we utilize data from the general election in 1993 to show how the current system can be used to the advantage of large parties. There is very little risk involved in running as two separate "parties" for a large party. However, it is not a good strategy for small parties because each "party" must attain at least 4% support at the national level to qualify for the apportionment of supplementary seats.

We use data from the general election in 1989 to devise the tactical plan for the general election in 1993 and pick out the parties to follow this plan. For a party to run as two "parties" without risking too much its support should be well above 10%. Only the two largest parties in 1989, A (Labour) with 34,3% and H (Conservatives) with 22,2%, meet this requirement. To determine which constituencies their second "parties" shall run in we identify constituencies which representation was more than 10 % below the national average in 1989. The 5 constituencies which satisfy this requirement are Akershus, Buskerud, Rogaland,

Oslo, and Østfold. This means that the first "parties" for A and H will run in the other 14 constituencies. We use election data from 1993 to evaluate the plan, i.e. we implicitly assume that the voters would have maintained their voting behaviour if the plan had been implemented. Given the described assumptions we get the apportionment of supplementary seats presented in Table 8.3. A1 and H1 denote the first "parties" for A and H respectively, while A2 and H2 denote the second "parties".

Table 8.3

Party lists	A1	A2	FrP	H1	H2	KrF	SP	SV
1st quotient	5940	**8183**	**11884**	6953	7508	7755	6341	**9268**
2nd quotient		**7835**	**10300**					**8462**
3rd quotient		7515	**9088**					7785
4th quotient			**8131**					
5th quotient			7357					
Seats	0	2	4	0	0	0	0	2

National quotients given the devised plan. Quotients in bold qualify for supplementary seats.

Before we comment on the apportionment in Table 8.3, let us reveal the real outcome of the general election in 1993: Neither A nor H got any supplementary seats. The supplementary seats were won by FrP (Progressives) with 4, SV (Socialists) with 3, and KrF (Christian democrats) with 1 seat. Now to the results of the tactical plan: From Table 8.3 we see that H2 does not win any supplementary seat, so H does not gain by running as two "parties". The situation for A is brighter. Its tactical split results in a gain of two supplementary seats compared with the real outcome. The losers are KrF and SV which lose one seat each. Here follows an explanation of why A2 wins supplementary seats: A2 only participates in constituencies which are underrepresented in preassigned (fixed) seats. Even if A2 has got its proportional share of seats in these constituencies, it is underrepresented compared with the national standard. That is why it ends up with supplementary seats. The purpose of supplementary seats is to even out disproportionality in the distribution of seats among parties, but the reality in A2's case is that the seats make up for the disproportional distribution of fixed seats among the constituencies. Finally, the changes in support from 1989 to 1993 might have contributed to A2's success and H2's lack of success. While H's

vote percentage decreased from 22,2% to 17,0%, A's increased from 32,3% to 36,9%.

8.5 House sizes in the Nordic countries

In this section we study the relationship between house size, population, and number of constituencies in the Nordic countries. By studying Table 2.3 in section 2.10 one gets the impression that the number of seats in a national parliament is influenced by the size of the population and the number of constituencies. Population size is presumably the most important influence. A rough rule seems to be that the house size increases with population, but less than proportionally. To study the suggested relationships more closely, we have gathered relevant data for the Nordic countries in Table 8.4 below. These data are taken from [Yearbook of Nordic Statistics 1994]. Our reason for focusing on the Nordic countries is that they have quite similar elections systems, which all are based on proportional representation. The nations in the lower part of Table 8.4 are territories with home rule.

Table 8.4

Country	Name	Seats	Population	Population/Seats	Constituencies	Seats/Constitiuencies
Sweden	Riksdag	349	8692013	24905	28	12,5
Finland *	Riksdag	199	5029989	25276	14	14,2
Denmark **	Folketing	175	5180614	29604	17	10,3
Norway	Storting	165	4299167	26056	19	8,7
Iceland	Althing	63	262386	4165	8	7,9
Faroe Islands **	Lagting	32	46804	1463	7	4,6
Åland *	Landsting	30	24993	833		
Greenland **	Landsting	27	55117	2041	18	1,5

* Åland has one seat in the Riksdag.

* * Faroe Islands and Greenland have two seats in the Folketing each.
These seats and constituencies are not included in the data for Finland and Denmark.

The Nordic parliaments at the end of 1992.

From time to time it is suggested that the number of seats in the Norwegian Storting should be reduced to get a more efficiently working parliament. A house size of about 100 has been mentioned. To evaluate this suggestion we look at the

ratios in Table 8.4. The national average population per seat in Norway is about the same as in the three other Nordic countries of comparable size, so an acceptance of the suggestion would lead Norway away from the Nordic norm. Moreover, the Norwegian seats per constituency ratio is already low compared with other Nordic countries. Let us analyse the consequences at the constituency level of reducing the house size to 100 seats. In this analysis we assume that the number of constituencies is left unchanged. The described reduction will reduce the seats per constituency ratio to about 5, which would make it even harder for small parties to win seats within the constituencies. The situation for V (Liberals) and RV (Red Electoral Alliance) in the general election in 1993 can serve as an example. In that election V and RV won one seat each, in Hordaland and Oslo respectively. Hordaland and Oslo with their 15 seats each are the two constituencies with most preassigned seats. The suggested reduction in the house size would have reduced their number of preassigned seats to about 9, in which case neither V nor RV would have been represented. Our conclusion is that the suggestion should be rejected. There is room for considerable improvement of the Norwegian election system, but the number of seats in the Storting is not the problem. If any of the Nordic countries should think about a reduction of the house size, it is Sweden. A house size of 349 seats is relatively high compared with other European countries of comparable size.

8.6 Motivation for the matrix problem

Chapters 1 through 8 have dealt with the vector apportionment problem, which represents a one-dimensional apportionment. However, in many cases we face a matrix of votes, where the votes represent both constituency and party. The question we now adress is how to carry out the apportionment in these situations.

A solution is to stay in the one-dimensional world and carry out several vector apportionments. Be aware that the succession of the apportionments has an impact on the final apportionment. It is normal to determine the constituency

apportionment first and thereafter apportion seats to the parties within each constituency. This succession of apportionments gives a proportional constituency representation plus proportional apportionments within the constituencies. The drawback is that the party representation at the national level often becomes disproportional. Contrary, if we start with the apportionment of seats among the parties, we get a proportional party representation. If we follow up this party apportionment by apportioning seats to the constituencies within each party, we will probably get a disproportional constituency representation. Thus, it is difficult to find a final apportionment which is proportional for both constituencies and parties.

Some countries have introduced supplementary seats to make the party representation at the national level less disproportional. A brief description of how the apportionment process is carried out in this case: First the number of fixed seats for each constituency is determined, either by law or apportionment. Within each constituency these seats are apportioned to the parties. The final part of the process is the apportionment of supplementary seats to the parties. This apportionment is based on the parties' national vote totals and takes the number of seats won in the constituencies into account. This way the supplementary seats repair for at least some of the disproportionality caused by the apportionments within the constituencies. The success of the repair process depends on how disproportional the initial party representation is and the number of supplementary seats. A large number of supplementary seats enhances the probability of a proportional party representation. Denmark, Sweden, Iceland, and Norway have 40, 39, 13, and 8 supplementary seats respectively, corresponding to 23%, 11%, 21%, and 5% of the house size. How proportional the constituency representation becomes after such an apportionment scheme depends on the way the supplementary seats are assigned to the constituencies. The constituency representation will be proportional if the supplementary seats are proportionally preassigned to the constituencies and the parties have to pick among these preassigned seats. However, the constituency representation may become disproportional if each supplementary seat is assigned freely to the constituency where the eligible party is most underrepresented. In Norway and

Sweden the supplementary seats are not preassigned to the constituencies, Denmark has a kind of partial preassignment, see [H-S] (page III.5), while Iceland after a law amendment in 1995 has all seats preassigned.

Apportionment schemes with supplementary seats represent an improvement compared with schemes without such seats. In part II of the dissertation we present an even better apportionment scheme, namely matrix apportionment. With this scheme the seats are apportioned simultaneously to constituency and party given predetermined constituency and party bounds. One of the advantages of matrix apportionment is that the bounds can be used to make both constituency and party representation proportional. Since a matrix apportionment takes both constituency and party into consideration at the same time, it is a two-dimensional apportionment. [B&D-1], [B&D-2], and [H&J] have shown how it can be carried out.

Part II:

The Matrix Apportionment Problem

Chapter 9: Introduction

The description of the matrix apportionment problem in this chapter builds on [B&D-1] and [B&D-2]. We begin the chapter by introducing basic terminology in section 9.1. In the second section we demonstrate that the free matrix apportionment problem is equivalent to the (free) vector apportionment problem. This is one of the reasons for introducing additional constraints in section 9.3. In section 9.4 we use the shaping of these constraints to distinguish between three types of matrix apportionment problems. Section 9.5 presents Balinski and Demange's axioms (conditions) for the matrix apportionment problem and tells which methods that satisfy them. The last two sections investigate the existence of solutions to the matrix allocation and matrix apportionment problem.

9.1 Basic terminology

The task in the general (two-dimensional) **matrix apportionment problem** is to distribute h seats among m constituencies and n parties based on the votes the parties have received in the constituencies. A (two-dimensional) matrix apportionment is an apportionment where each seat represents both constituency and party. h, m, and n are positive integers. Notice that election situations with $m = 1$ or $n = 1$ are nothing but the vector apportionment problem. We use the indexes i and j for constituencies and parties respectively, where $i \in M = \{1, 2, ..., m\}$ and $j \in N = \{1, 2, ..., n\}$. $\mathbf{p} = (p_{ij})$ is the **vote matrix** (population matrix), where p_{ij} denotes the votes for party j in constituency i. We call each entry in a matrix, i.e. each combination of constituency and party, **cell** and refer to it by its positioning in the matrix. Every cell in a vote matrix is assumed to be a non-negative real

number, i.e. $0 \leq p_{ij} \in \Re$ for all i, j. The reason for allowing some cells to be zero is that there are parties which only participate in one or a few constituencies. However, we do not allow parties with zero national vote totals or constituencies with no voters. Should such a party or constituency appear, we simply eliminate it from the election situation. A matrix problem is called **positive** if all cells in the vote matrix are positive, i.e. if $\mathbf{p} > \mathbf{0}$. The **apportionment matrix** is denoted $\mathbf{a} = (a_{ij})$, where all cells a_{ij} are non-negative integers, i.e. $\mathbf{a} \geq \mathbf{0}$. Both $\mathbf{p}$ and $\mathbf{a}$ are $m \times n$ matrices. We use $\mathbf{a}$ to exemplify the notation of sums of cells: a_{iN} and a_{Mj} denote the number of seats apportioned to constituency i and party j respectively:

$$a_{iN} = \sum_{j \in N} a_{ij} \qquad a_{Mj} = \sum_{i \in M} a_{ij} \tag{9.1}$$

a_{MN} is the total number of seats apportioned. Moreover, the house size h can be expressed in several ways:

$$\sum_{i \in M} \sum_{j \in N} a_{ij} = \sum_{i \in M} a_{iN} = \sum_{j \in N} a_{Mj} = a_{MN} = h \tag{9.2}$$

9.2 The free matrix apportionment problem

A matrix apportionment problem is **free** if the house size constraint (9.2) is the only constraint. The vector apportionment problem, as presented in chapters 1 and 6, is free. On the other hand, a problem where minimum representations fixed by law are present is not free. Even if $\mathbf{p}$ is a $m \times n$ matrix in the matrix problem compared to a vector of length m or n in the vector problem, the two free problems are essentially the same. We now explain how the free matrix problem can been converted to an equivalent free vector problem: There are no other restrictions on a_{iN} and a_{Mj} than the one imposed indirectly through the house size constraint (9.2). The vote matrix can therefore be rearranged. We choose to split it up in m vectors of length n and place these vectors one after another. The result of this rearrangement is a vector of length $m \cdot n$, where

element number $(s - 1)\cdot m + a$, where $s \in M$ and $a \in N$, is the votes of the ath party in the sth constituency.

Each cell is on its own in the free matrix apportionment problem. A cell representing the same party or constituency is just as fierce an opponent as any other cell. Thus, the vote totals for constituencies and parties are irrelevant for the apportionment. As a result, the free matrix apportionment problem will often lead to a disproportional representation for both constituencies and parties. One of the reasons for having parties must be that their lists in different constituencies are seen as a part of a whole apportionmentwise. By this we mean that if a party just has missed out in the competion for seats in some constituencies, this should count in its favour if there is a close race in another constituency. Regarding the constituencies, we stated in chapter 8 that their representation should be proportional to some chosen basis. Our conclusion after this brief discussion is that it is a bad solution to let the matrix apportionment problem be free. In fact, most, if not all, of the apportionment schemes described in section 8.6 are better.

9.3 Additional constraints for the matrix problem

To accomplish a higher degree of proportionality between vote totals and apportionment than the free matrix problem could guarantee, we introduce constituency and party constraints. We call a problem **equality constrained** if all constituency and party constraints are given as equalities. Later on we operate with equality constraints, but to be general we here present the inequality versions of the constituency (9.3) and party constraints (9.4):

$$r_i \le a_{iN} \le R_i \qquad \forall i \tag{9.3}$$

$$c_j \le a_{Mj} \le C_j \qquad \forall j \tag{9.4}$$

r_i and c_j are lower, while R_i and C_j are upper bounds. $\mathbf{r} = (r_i)$ and $\mathbf{R} = (R_i)$, which both are vectors of length m, summarize the bounds for the constituency

representation. Similarly, the party bound vectors are $\mathbf{c} = (c_j)$ and $\mathbf{C} = (C_j)$ of length n. We assume that all lower bounds are non-negative integers and that all upper bounds are positive integers:

(9.5) $\mathbf{r} \geq 0$ and $\mathbf{c} \geq 0$

(9.6) $\mathbf{R} > 0$ and $\mathbf{C} > 0$

We do not exclude any interesting situations by assuming that the upper bounds are larger than zero. If an upper bound of zero had been allowed, the constituency or party associated with this constraint would not have got any seat and could therefore be eliminated from the problem. For the lower and upper bounds to be consistent with the total number of seats, the following relationships must hold for constituencies and parties respectively:

(9.7) $$r_M = \sum_{i \in M} r_i \leq h \leq \sum_{i \in M} R_i = R_M$$

(9.8) $$c_N = \sum_{j \in N} c_j \leq h \leq \sum_{j \in N} C_j = C_N$$

How shall the bounds be determined? In section 8.6 we argued that both constituency and party representation should be proportional. When the problem is equality constrained this is easily achieved by carrying out two vector (one-dimensional) apportionments: To determine proportional constituency bounds we apportion seats to the constituencies based on the number of people entitled to vote or some appropriate constructed measure. The party bounds are determined by apportioning seats to the parties based on their national vote totals. In the determination of constituency and party bounds the arguments from chapter 8 regarding choice of apportionment method should be taken into account. Even for problems which are not equality constrained it seems reasonable to determine the bounds by means of apportionment methods. These apportionments should be based on house sizes of r_M, c_N, etc, where r_M and c_N are not necessarily equal. In countries with "free" supplementary seats the upper constituency bounds are often set to infinity. However, the house size constraint (9.2) restricts the real upper bounds to $R_i = r_i + (h - r_M)$.

All parameters for the matrix apportionment problem have now been introduced, and we gather them in the **bound vector**: $\sigma = (\mathbf{c}, \mathbf{C}, \mathbf{r}, \mathbf{R}, h)$. Even if we had allowed $\mathbf{c}$, $\mathbf{C}$, $\mathbf{r}$, $\mathbf{R}$, and h to be real, all elements in σ would effectively be integer because the apportionment matrix is integer. This is the motivation for the earlier integer assumption regarding the bounds. We call a pair $(\mathbf{p},\sigma)$ an **election situation** for the matrix apportionment problem.

The kind of problem we get when allowing both the apportionment matrix and the bound vector to be real valued is called **allocation problem**. We call the real valued version of the apportionment matrix **allocation matrix** and denote it $\mathbf{f} = (f_{ij})$, where $f_{ij} \in \Re$ for all i, j. The allocation problem has the same constraints as the apportionment problem. Constraints (9.2) - (9.4) with f_{ij} substituted for a_{ij} determine the **set of allocations**. We denote this set $R(\sigma)$:

$$R(\sigma) = \{\mathbf{f} = (f_{ij}) \geq 0: r_i \leq f_{iN} \leq R_i,\ c_j \leq f_{Mj} \leq C_j,\ f_{MN} = h\} \tag{9.9}$$

The **set of apportionments** consists of the integer valued members of $R(\sigma)$. From now on we assume, unless otherwise stated, that we deal with problems where the set of apportionments is non-empty. As in the vector case we need an **apportionment method** A to determine the apportionment(s). A method used to determine allocations is called **allocation method** and is denoted F. The allocation problem is also known as the **continuous problem**, while the **integer problem** is another name for the apportionment problem.

9.4 Types of matrix apportionment problems

We distinguish between three types of matrix apportionment problems: The equality constrained problem, the partial inequality constrained problem, and the inequality constrained problem. An **inequality constrained problem** is a problem where both constituency and party constraints are given as inequalities and where $r_M < h$, $c_N < h$, $R_M > h$, and $C_N > h$. The free matrix apportionment

problem is a special case of the inequality constrained problem, namely a problem where all lower bounds are set equal to zero and all upper bounds are set equal to infinity. With a **partial inequality constrained problem** we mean a problem with equality constraints for one index and inequality constraints for the other. Election systems with free supplementary seats, like the Swedish and Norwegian ones, can be viewed as this type of problem. Interpreted in a matrix context such systems have equality party constraints because $c_N = C_N = h$, but inequality constituency constraints because $r_M < h$ and $R_M > h$.

In the choice between the different ways of formulating the matrix apportionment problem we prefer the equality constrained version. There is no need for inequality party constraints because the party representation should be proportional to the parties' national vote totals. In section 8.2 we argued that the constituency representation should be proportional to a constructed measure which includes the number of eligible voters. The drawback of the proposed constructed measure is that it does not take the number of votes cast into consideration. However, if the number of votes cast is included as an important part of such a constructed measure, there should not be a need for inequality constituency constraints. NB. In the tests in sections 11.5 and 12.4 we use equality constituency constraints determined from the vote totals.

9.5 Balinski and Demange's axioms

[B&D-1] (page 711) propose six axioms for the matrix apportionment problem. The first five of these are directly inspired by their five allocation axioms, see [B&D-1] (page 702), while the sixth axiom is special for the apportionment problem. Because of the similarity we only present the six apportionment axioms here. Five of them are generalizations of conditions from the vector apportionment problem, and we use the same names.

Regarding **exactness**, the only distinction compared with the vector apportionment problem is that instead of using quotas we here use fair shares f_{ij}^*. The fair

share matrix $\mathbf{f}^*$ deviates from the proportional **quota matrix** $\mathbf{q} = (q_{ij}) = (\frac{h \cdot p_{ij}}{p_{MN}})$ via row and column multipliers which take the constituency and party bounds into consideration.

Definition 9.1
A is **exact** if $f^*_{ij} \in N_0$ for all i, j implies that $A(\mathbf{p},\sigma) = \mathbf{f}^* = (f^*_{ij})$.

For positive problems N can be substituted for N_0. The probability of all cells in a fair share matrix being integer is extremely small.

Relevance is the new axiom (condition) compared with the vector apportionment problem. [B&D] see it as a kind of "independence of irrelevant alternatives" property. It says that if some of the apportionments of $A(\mathbf{p},\sigma)$ belong to the more restricted region $R(\hat{\sigma})$, then one cannot obtain a better set of apportionments $A(\mathbf{p},\hat{\sigma})$ than those inside this region. Thus, the possibilities in $R(\sigma)$ - $R(\hat{\sigma})$, i.e. outside the restricted region $R(\hat{\sigma})$, should be irrelevant.

Definition 9.2
A is **relevant** if the following holds for all $(\mathbf{p},\sigma)$ and $(\mathbf{p},\hat{\sigma})$:
If $[A(\mathbf{p},\sigma) \cap R(\hat{\sigma})] \neq \varnothing$ and $R(\hat{\sigma}) \subset R(\sigma)$, then $A(\mathbf{p},\hat{\sigma}) = [A(\mathbf{p},\sigma) \cap R(\hat{\sigma})]$.

Before the next axiom is presented, we need to introduce notation for submatrices. As an example, the submatrix of $\mathbf{p}$ defined over the constituencies $I \subset M$ and parties $J \subset N$ is denoted $\mathbf{p}_{IJ}$.

The idea behind **consistency** is that the apportionment for any subproblem shall be consistent with the apportionment for the whole problem. [B&D-1] (page 711) call this axiom **uniformity**.

Definition 9.3
A is **consistent** if $\mathbf{a} \in A(\mathbf{p},\sigma)$ implies that $\mathbf{a}_{IJ} \in A(\mathbf{p}_{IJ},\sigma_{IJ})$;
and if $\hat{\mathbf{a}}_{IJ} \in A(\mathbf{p}_{IJ},\sigma_{IJ})$ is another apportionment for the subproblem,
then $\hat{\mathbf{a}}$ defined to be equal to $\hat{\mathbf{a}}_{IJ}$ on $I \times J$ and $\mathbf{a}$ elsewhere
is also an apportionment: $\hat{\mathbf{a}} \in A(\mathbf{p},\sigma)$.

The counterpart to the condition weak population monotonicity from the vector apportionment problem is called **monotonicity**. If the number of votes increases for only one cell, monotonicity demands that this cell shall not lose any seat.

Definition 9.4
A is **monotone** if $\mathbf{a} \in A(\mathbf{p},\sigma)$, $\hat{\mathbf{a}} \in A(\hat{\mathbf{p}},\sigma)$, and $\hat{\mathbf{p}}$ is equal to $\mathbf{p}$ except for $p_{sa} < \hat{p}_{sa}$, where $s \in M$ and $a \in N$, imply $a_{sa} \leq \hat{a}_{sa}$.

Let us define the following sets: $I^- = \{i \in M: a_{iN} = r_i\}$, $I^0 = \{i \in M: r_i < a_{iN} < R_i\}$, $I^+ = \{i \in M: a_{iN} = R_i\}$ and analogously for J^-, J^0, and J^+.

Due to the constituency and party bounds, the axiom (condition) called **homogeneity** is more complex here than it was for the vector apportionment problem because of the constituency and party bounds. Homogeneity now requires the following: If two rows or columns are proportional and bounded to the same sum, then there should be an apportionment that is equal for these rows or columns.

Definition 9.5
A is **homogeneous** if $\mathbf{a} \in A(\mathbf{p},\sigma)$ and $\delta > 0$, $\alpha > 0$, and $\beta > 0$ are such that

$\alpha_i < 1$ implies $i \in I^-$ and $\alpha_i > 1$ implies $i \in I^+$,
$\beta_j < 1$ implies $j \in J^-$ and $\beta_j > 1$ implies $j \in J^+$,

imply that $\mathbf{a} \in A(\delta \cdot \alpha \cdot \mathbf{p} \cdot \beta, \sigma)$.

Completeness is the axiom which is special for the apportionment problem, because ties do not occur in the allocation problem. It is defined the same way as for the vector apportionment problem:

Definition 9.6
A is **complete** if $\mathbf{p}^a \to \mathbf{p}$ when a tends to infinity and $\mathbf{a} \in A(\mathbf{p}^a,\sigma)$ for every a, then $\mathbf{a} \in A(\mathbf{p},\sigma)$.

[B&D-1] (page 711) let A^d denote a divisor method which satisfies Definition 2.2. A divisor rounding $[\ ]_d$ in Definition 9.7 below is defined the same way as for the vector apportionment problem, see equation (2.10) in section 2.7. What is different here is that we have three multipliers to adjust the votes $\delta \cdot \lambda_i \cdot p_{ij} \cdot \mu_j$ compared to one $\frac{1}{z} \cdot p_i = \delta \cdot p_i$ for the free vector problem. λ and μ are needed to obey the constituency and party bounds respectively:

Definition 9.7

a is an apportionment for the election situation $(\mathbf{p},\sigma)$ with the divisor method A^d if:

$$\mathbf{a} = (a_{ij}) = ([\delta \cdot \lambda_i \cdot p_{ij} \cdot \mu_j]_d), \quad \mathbf{a} \in R(\sigma),$$

for some choice of $\delta > 0$, $\lambda > 0$, $\mu > 0$ which also satisfy

$$\lambda_i > 1 \text{ implies } a_{iN} = r_i \quad \text{and} \quad \lambda_i < 1 \text{ implies } a_{iN} = R_i,$$
$$\mu_j > 1 \text{ implies } a_{Mj} = c_j \quad \text{and} \quad \mu_j < 1 \text{ implies } a_{Mj} = C_j.$$

The possibility of ties means that the chosen divisor method A^d may admit several apportionments for a given election situation. A multiplier set (δ,λ,μ) which satisfies the premises in Definition 9.7 is called **proper** for **a** in $A^d(\mathbf{p},\sigma)$. There is usually some freedom in the choice of multiplier values since many different adjusted votes $\delta \cdot \lambda_i \cdot p_{ij} \cdot \mu_j$ result in the same divisor rounding a_{ij}. The value of a multiplier not involved in a tie can be chosen within an interval given fixed values of the other multipliers. Thus, there are usually infinitely many sets of proper multipliers for $(\mathbf{p},\sigma)$ given A^d.

[B&D-1] state the following two characterization theorems, which they prove on (page 713→) and (page 705) respectively:

Theorem 9.1

An apportionment method satisfies the six apportionment axioms over the class of positive problems if and only if it is a divisor method A^d with $d_1 > 0$.

Theorem 9.2
The **fair share method** F^* is the unique matrix allocation method which satisfies the five allocation axioms over the class of positive problems.

The fair share method produces what is called a **fair share matrix**. This matrix is denoted $\mathbf{f}^*$ and can be expressed as $\mathbf{f}^* = (f_{ij}^*) = (\delta \cdot \lambda_i \cdot p_{ij} \cdot \mu_j)$, where the multipliers satisfy the premises in Definition 9.7 with f_{ij}^* substituted for a_{ij}. For a given δ value there is usually less leeway regarding the choice of multiplier values for the allocation problem than for the corresponding apportionment problem. This is not the case in the following situation:

Example 9.1
Consider the election situation with $\mathbf{p} = \begin{vmatrix} 2 & 2 \\ 1 & 1 \end{vmatrix}$, $\mathbf{R} = (1, 1)$, $\mathbf{C} = (1, 1)$, and $h = 2$, i.e. an equality constrained problem. We choose MF ($CP_{\frac{1}{2}}$) as the apportionment method and let $\delta = \frac{h}{p_{MN}} = \frac{1}{3}$. Then $\lambda_1 = \frac{3}{4}$, $\lambda_2 = \frac{3}{2}$, $\mu_1 = \mu_2 = 1$ is the only proper multiplier set. It can be shown that the fair share method also produces these multiplier values given $\delta = \frac{1}{3}$. In both instances the final adjusted vote matrix $\delta \cdot \lambda \cdot \mathbf{p} \cdot \mu$ is as shown in Table 9.1. This is also the optimal allocation, while MF allows both $\begin{vmatrix} 1 & 0 \\ 0 & 1 \end{vmatrix}$ and $\begin{vmatrix} 0 & 1 \\ 1 & 0 \end{vmatrix}$ as apportionments.

Table 9.1

$$\begin{vmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{vmatrix}$$

Final adjusted vote matrix.

9.6 Existence of matrix allocations

We start the review of existence results for the matrix allocation problem with positive problems, i.e. problems with $\mathbf{p} > 0$. An immediate consequence of Theorem 9.3 below is that there exists a unique solution to all positive allocation problems.

Difficulties regarding existence arise when we face vote matrices with zeros. $R(\sigma)$ is too general for our purpose in these situations. Since it is unreasonable for a cell with zero votes to get a positive allocation, we can restrict our attention to the following subset of $R(\sigma)$:

(9.10) $$R^0(\mathbf{p},\sigma) = \{\mathbf{f} \in R(\sigma): f_{ij} = 0 \text{ if } p_{ij} = 0\}$$

Due to limitations of the software used to write this dissertation, we denote the complements of $I \subset M$ and $J \subset N$ by $\underline{I}$ and $\underline{J}$ respectively. In the following we assume that the $I \times \underline{J}$ subset of the vote matrix consists entirely of zeros, i.e. $p_{I\underline{J}} = 0$. Then necessary and sufficient conditions for $R^0(\mathbf{p},\sigma)$ to be non-empty are the following adaption of the supply-demand conditions of network flow theory, see [B&D-1] (page 707):

(9.11) $$C_J \geq r_I, \qquad R_{\underline{I}} \geq c_{\underline{J}}, \qquad \text{and} \qquad c_{\underline{J}} + r_I \leq h \leq C_J + R_{\underline{I}}$$
for any $I \subset M$ and $J \subset N$ with $p_{I\underline{J}} = 0$ ($r_\varnothing = 0$ etc).

It is tempting to demand even more for allocation problems; namely that the allocation matrix only contains zeros if there are corresponding zeros in the vote matrix. The subset of $R^0(\mathbf{p},\sigma)$ thereby defined is:

(9.12) $$R^+(\mathbf{p},\sigma) = \{\mathbf{f} \in R(\sigma): f_{ij} = 0 \text{ if and only if } p_{ij} = 0\}$$

$R^+(\mathbf{p},\sigma) \subset R^0(\mathbf{p},\sigma) \subset R(\sigma)$. The conditions in (9.11) regarding the non-emptiness of $R^0(\mathbf{p},\sigma)$ are not enough to guarantee that $R^+(\mathbf{p},\sigma)$ is non-empty. [B&D-1] (page 708) and [B&D-2] (page 198) present further definitions which are of help in characterizing when $R^+(\mathbf{p},\sigma) \neq \varnothing$:

An election situation $(\mathbf{p},\sigma)$ which satisfies the conditions in (9.11) is said to be **reducible** into independent subproblems on (I, J) and $(\underline{I}, \underline{J})$ respectively if I and J are proper subsets of M and N for which an inequality in (9.11) is binding. If also $p_{I\underline{J}} = 0$, $(\mathbf{p},\sigma)$ is said to be **decomposable** into independent subproblems on (I, J) and $(\underline{I}, \underline{J})$. The conditions the different subproblems must satisfy depend on

which of the inequalities in (9.11) that is satisfied as an equality. When $C_J = r_I$, the subproblem on $(\underline{I}, \underline{J})$ must sum to h - C_J, while if $R_{\underline{I}} = c_{\underline{J}}$, the subproblem on (I, J) must sum to h - $R_{\underline{I}}$. Moreover, if $C_J + R_{\underline{I}} = h$, the subproblem on (I, J) must sum to C_J while that on $(\underline{I}, \underline{J})$ must sum to $R_{\underline{I}}$; and similarly if $c_{\underline{J}} + r_I = h$. Finally, $(\mathbf{p},\sigma)$ is said to be **irreducible** if all inequalities in (9.11) are satisfied strictly, i.e. $C_J > r_I$, $R_{\underline{I}} > c_{\underline{J}}$, and $c_{\underline{J}} + r_I < h < C_J + R_{\underline{I}}$, whenever the subsets are proper.

[B&D-2] (page 198) prove that $R^+(\mathbf{p},\sigma) \neq \varnothing$ if and only if $(\mathbf{p},\sigma)$ is decomposable into a set of independent irreducible subproblems. We are now ready for the principal result regarding existence of matrix allocations, [B&D-2] (page 197):

Theorem 9.3
A fair share matrix exists if and only if $R^+(\mathbf{p},\sigma)$ is non-empty, in which case it is the unique solution.

[B&D-1] (page 707) define a matrix $\mathbf{f}$ as an **extended fair share matrix** for the election situation $(\mathbf{p},\sigma)$ if there exists some sequence of election situations $(\mathbf{p}^a,\sigma^a)$, $\mathbf{p}^a > 0$ converging to $(\mathbf{p},\sigma)$ such that the sequence $F^*(\mathbf{p}^a,\sigma^a)$ converges to $\mathbf{f}$. When $R^0(\mathbf{p},\sigma) = \varnothing$, the extended fair share matrix does not give a satisfactory solution to the allocation problem, as exemplified in [B&D-2] (page 199-200). However, the extended fair share matrix is useful in election situations where $R^+(\mathbf{p},\sigma) = \varnothing$ but $R^0(\mathbf{p},\sigma) \neq \varnothing$ because of the following theorem from [B&D-1] (page 707):

Theorem 9.4
If the set $R^0(\mathbf{p},\sigma)$ is non-empty, then the extended fair share matrix of the election situation $(\mathbf{p},\sigma)$ is unique. Moreover, it belongs to $R^0(\mathbf{p},\sigma)$.

The unique fair share matrix is instrumental in the calculation of matrix bias in chapter 16. In section 16.2 we present the standard algorithm for determining this matrix.

9.7 Existence of matrix apportionments

It follows from Theorem 9.1 that there exist solutions to all positive matrix apportionment problems whenever $d_1 > 0$. Unlike for the allocation problem, we see nothing wrong in not apportioning any seat to a cell with a positive number of votes. However, if one demands that all such cells shall qualify for a positive apportionment, the following subset of $R^0(\mathbf{p},\sigma)$ is of interest:

$$R^1(\mathbf{p},\sigma) = \{\mathbf{f} \in R^0(\mathbf{p},\sigma): p_{ij} > 0 \text{ implies } a_{ij} \geq 1\} \tag{9.13}$$

$R^1(\mathbf{p},\sigma)$ is the apportionment counterpart to $R^+(\mathbf{p},\sigma)$, but the integer requirement makes it more demanding since all cells with a positive number of votes, however small, must get at least one seat. [B&D-1] (page 712) present the following theorem regarding existence of matrix apportionments:

Theorem 9.5
If $d_1 > 0$, $A^d(\mathbf{p},\sigma)$ is non-empty if and only if $R^0(\mathbf{p},\sigma)$ is non-empty, while with $d_1 = 0$, $A^d(\mathbf{p},\sigma)$ is non-empty if and only if $R^1(\mathbf{p},\sigma)$ is non-empty.

The number of positive vote cells cannot be larger than h for $R^1(\mathbf{p},\sigma)$ to be non-empty. For many real world election situations this requirement is not satisfied, so an A^d with $d_1 = 0$ should not be used.

The existence results from this and the preceding section tell us that we will avoid trouble if $R^0(\mathbf{p},\sigma)$ is non-empty. It is therefore interesting to know whether $R^0(\mathbf{p},\sigma)$ may be empty with our recommendations regarding determination of constituency and party bounds. Based on the discussion in sections 8.1 and 8.3 we recommend that a divisor method used to determine constituency (party) bounds from vote totals should not be less (more) favourable to small entities than MF. In our empirical test in chapter 14 we operate with two combinations of bounds determined from vote totals: The first combination is constituency bounds determined by SD together with party bounds determined by HA, while the

second combination is both types of bounds determined by MF. As the two examples below illustrate, it is possible that $R^0(\mathbf{p},\sigma) = \varnothing$ with these bound combinations. Both examples deal with equality constrained problems, so we denote the constituency and party bounds **R** and **C** respectively.

Example 9.2

We face the 2 × 2 vote matrix in Table 9.2 with a house size of 7. Based on the vote totals we determine the constituency bounds by SD and the party bounds by HA. The resultant bounds are **R** = (5, 2) and **C** = (6, 1) respectively. Since the apportionment to the cell in the upper left corner a_{11} is restricted to a maximum of 5 seats by the constraint for the first constituency $R_1 = 5$, the apportionment to the cell with zero votes in the lower left corner a_{21} must be at least 1 seat to satisfy the constraint for the first party $C_1 = 6$. Hence, $R^0(\mathbf{p},\sigma)$ is empty. This can also be verified through (9.11). Because $p_{21} = 0$, $I = \{2\}$ and $J = \{1\}$. The first inequality in (9.11) requires that $C_2 \geq R_2$, but $1 = C_2 < R_2 = 2$ here. Moreover, the two other conditions in (9.11) are also violated.

Table 9.2

37	2	39
0	10	10
37	12	

2 × 2 vote matrix and its row and column vote totals.

Table 9.3

412	10	6	428
0	125	75	200
412	135	81	

2 × 3 vote matrix and its row and column vote totals.

With MF to determine both types of bounds, a 2 × 2 vote matrix is not enough to construct an election situation where $R^0(\mathbf{p},\sigma) = \varnothing$. However, with a 2 × 3 vote matrix it is possible that $R^0(\mathbf{p},\sigma)$ is empty:

Example 9.3

We face an election situation consisting of the vote matrix given in Table 9.3 and a house size of 11. Both constituency and party bounds are determined by MF based on the vote totals, which result in the bounds **R** = (7, 4) and **C** = (8, 2, 1) respectively. Like in Example 9.2 these bounds require that the apportionment to the cell with zero votes a_{21} shall be at least 1 seat. Thus, once again $R^0(\mathbf{p},\sigma) = \varnothing$.

The examples above are not that unrealistic, so it is not too promising that they lead to an empty $R^0(\mathbf{p},\sigma)$. Nevertheless, here follows some arguments against their realism: Both examples are constructed such that both the constituency and the party the cell with zero votes belongs to are rather fortunate in the apportionments to determine bounds. Moreover, a cell with zero votes often belongs to a small party, while it here belongs to the largest party. The arguments above point to a low probability of $R^0(\mathbf{p},\sigma)$ being empty. During the algorithm runs in connection with the tests in chapters 14 and 16 we did not encounter any occurrence of $R^0(\mathbf{p},\sigma) = \varnothing$.

Chapter 10: Formulations

In this chapter we look at two constrained optimization formulations of the matrix apportionment problem. The first two sections follow [H&J]. We start off by presenting the entropy minimization formulation. This formulation has a dual which is analysed in section 10.2. In the last section the utility maximization formulation is presented.

10.1 Entropy formulation

It has been known for a while that the positive equality constrained matrix allocation problem can be formulated as a non-linear optimization problem where an entropy function is minimized, see [Ba] (page 83). [B&D-2] (page 196) use entropy minimization also for the inequality constrained matrix allocation problem. To handle the difficulties which arise because of cells with zero votes, they introduce the following notation: S is the set of cells for which $p_{ij} > 0$, while $\bar{S}$ is its complement, i.e. the set of cells for which $p_{ij} = 0$. The formulation of the problem then becomes:

$$\min \sum_{(i,j)\in S} f_{ij} \cdot [\ln(\frac{f_{ij}}{p_{ij}}) - 1] \tag{10.1}$$

subject to the constraints

$$r_i \le \sum_{j\in N} f_{ij} = f_{iN} \le R_i \qquad \forall\, i \tag{10.2}$$

$$c_j \le \sum_{i\in M} f_{ij} = f_{Mj} \le C_j \qquad \forall\, j \tag{10.3}$$

$$\sum_{i \in M} \sum_{j \in N} f_{ij} = f_{MN} = h \tag{10.4}$$

$$f_{ij} \geq 0 \qquad \forall (i,j) \in S \tag{10.5}$$

$$f_{ij} = 0 \qquad \forall (i,j) \in \bar{S} \tag{10.6}$$

The feasible set for this program is $R^0(\mathbf{p},\sigma)$. By using Lagrangian relaxation on the formulation above it can be shown that the unique optimal solution when $R^+(\mathbf{p},\sigma) \neq \varnothing$ can be expressed as:

$$f^*_{ij} = p_{ij} \cdot e^{(\alpha_i - A_i + \beta_j - B_j + \delta)} \qquad \text{for } (i,j) \in S \tag{10.7}$$

$$f^*_{ij} = 0 \qquad \text{for } (i,j) \in \bar{S}$$

where α_i and A_i are related to the constituency constraint (10.2), β_j and B_j to the party constraint (10.3), and δ is related to the total seats equality (10.4).

Entropy is a measure of the order of chaos. The larger the value of an entropy function, the greater the order of chaos. Our goal is to make the apportionment matrix **a** proportional to the vote matrix **p** subject to the requirement that the number of seats distributed to each constituency and each party fall within specified limits. For this sake we define the entropy function π_{ij} which measures the order of chaos between **a** and **p**. We assume that π_{ij} is continuous, monotonically increasing, and strictly convex. To obtain the proportional apportionment we minimize the value of the entropy function:

$$\min \sum_{i \in M} \sum_{j \in N} \pi_{ij}(a_{ij}) \tag{10.8}$$

The constrained optimization problem with (10.8) as objective function is a non-linear integer programming problem. Since we only are interested in the value of π_{ij} for integer arguments, we linearize π_{ij}. The linearization is accomplished by introducing the 0/1-variables a_{ijl} and the coefficients π_{ijl}. a_{ijl} relates to a_{ij} through $\sum_{l \in H} a_{ijl} = a_{ijH} = a_{ij}$. $l \in H$ is here the seat number in cell (i,j) and therefore a member of the possibly more restricted set $\{1, 2, ..., \min(R_i, C_j)\}$. We define the coefficient π_{ijl} as the increment from $\pi_{ij}(l - 1)$ to $\pi_{ij}(l)$:

(10.9) $$\pi_{ij1} = \pi_{ij}(1)$$
$$\pi_{ijl} = \pi_{ij}(l) - \pi_{ij}(l-1) \qquad \text{for } l \geq 2$$

The fact that $\pi_{ij}(a_{ij})$ is monotonically increasing and strictly convex ensures that π_{ijl} forms a strictly increasing sequence in l, i.e. $\pi_{ijl} > \pi_{ij(l-1)}$ for $l \geq 2$. After the linearization, the objective function looks like this:

(10.10) $$\min \sum_{i \in M} \sum_{j \in N} \sum_{l \in H} [\pi_{ijl} \cdot a_{ijl}]$$

We now relate the coefficients π_{ijl} to divisor methods by letting $\pi_{ijl} = \ln(\frac{d_l}{p_{ij}})$. Explanation of why this is a suitable choice: Since the quotients $\frac{p_{ij}}{d_l}$ form a strictly decreasing sequence in l for all strict divisor methods, their reciprocals $\frac{d_l}{p_{ij}}$ form a strictly increasing sequence in l. The use of the natural logarithm only represents a monotonic transformation of this strictly increasing sequence. For integer arguments the entropy function can now be expressed as:

(10.11) $$\pi_{ij}(l) = \sum_{s=1}^{l} \pi_{ijs} = \ln(\frac{d_1}{p_{ij}}) + \ln(\frac{d_2}{p_{ij}}) + \ldots + \ln(\frac{d_l}{p_{ij}}) = \ln(d_l!) - l \cdot \ln(p_{ij})$$

For more comments on the entropy formulation we refer to [H&J] (page 7-13). The substitution of $\ln(\frac{d_l}{p_{ij}})$ for π_{ijl} converts (10.10) to:

(10.12) $$\min \sum_{i \in M} \sum_{j \in N} \sum_{l \in H} [\ln(\frac{d_l}{p_{ij}}) \cdot a_{ijl}]$$

which further is equivalent to the following objective function:

(10.13) $$\max \sum_{i \in M} \sum_{j \in N} \sum_{l \in H} [\ln(\frac{p_{ij}}{d_l}) \cdot a_{ijl}]$$

Because division by zero is undefined and so is the natural logarithm of 0, the instances where $d_1 = 0$ or $p_{ij} = 0$ need special attention. We treat $p_{ij} = 0$ first. Since a cell with zero votes should not get any seat, we remove all terms for which $(i,j) \in \bar{S}$ from the objective function and introduce the extra constraint (10.21) instead. Regarding $d_1 = 0$, we define $\ln(\frac{p_{ij}}{0})$ to be a very large but countable number. Moreover, we define the internal relationship between

different such quotients the same way as in Definition 2.1. After this clarification we are ready for what will be our constrained optimization formulation of the matrix apportionment problem:

$$\max \sum_{(i,j)\in S} \sum_{l\in H} [\ln(\frac{p_{ij}}{d_l}) \cdot a_{ijl}] \tag{10.14}$$

subject to the constraints

$$\sum_{i\in M} \sum_{j\in N} \sum_{l\in H} a_{ijl} = a_{MNH} = h \qquad \ln(\delta) \tag{10.15}$$

$$\sum_{j\in N} \sum_{l\in H} a_{ijl} = a_{iNH} \geq r_i \qquad \forall\, i \qquad \ln(\alpha_i) \tag{10.16}$$

$$\sum_{j\in N} \sum_{l\in H} a_{ijl} = a_{iNH} \leq R_i \qquad \forall\, i \qquad \ln(A_i) \tag{10.17}$$

$$\sum_{i\in M} \sum_{l\in H} a_{ijl} = a_{MjH} \geq c_j \qquad \forall\, j \qquad \ln(\beta_j) \tag{10.18}$$

$$\sum_{i\in M} \sum_{l\in H} a_{ijl} = a_{MjH} \leq C_j \qquad \forall\, j \qquad \ln(B_j) \tag{10.19}$$

$$a_{ijl} = 0 \text{ or } 1 \qquad \forall\, (i,j) \in S,\, l \tag{10.20}$$

$$a_{ijl} = 0 \qquad \forall\, (i,j) \in \bar{S},\, l \tag{10.21}$$

(10.15) - (10.19) express the same as (9.2) - (9.4), but both the constituency and party constraints have been splitted in two here. Be aware that a_{iNH} here is the same as a_{iN} in (9.3) etc.

We observe that the optimization problem above is linear in the variables a_{ijl}. The restriction that a_{ijl} has to be integer, actually zero or one, means that the problem is an integer programming problem. In fact, it is a network flow problem with integer capacity bounds σ. This means that if the integer restriction on a_{ijl} is loosened, the corresponding linear programming problem will have an optimal solution which is integer, see [E&G&S] (page 398). Let us therefore relax constraint (10.20) the following way:

$$a_{ijl} \geq 0 \qquad \forall\, (i,j) \in S,\, l \tag{10.22}$$

$$a_{ijl} \leq 1 \qquad \forall\, (i,j) \in S,\, l \qquad \ln(\gamma_{ijl}) \tag{10.23}$$

The constraint set consisting of (10.15) - (10.19) and (10.21) - (10.23) limits the set of possible apportionments to integer members of $R^0(\mathbf{p},\sigma)$. The variables to the right of the constraints above are the associated dual variables. These variables are utilized in the dual formulation in the next section.

10.2 The Dual Problem

A way of gaining insight into a primal problem is by formulating its dual. [E&G&S] (page 182-183) describe the formulation process. We are interested in the dual of the LP problem in section 10.1. Given the dual variables introduced above, the formulation of the dual problem becomes:

$$(10.24) \qquad \min \; -h \cdot \ln(\delta) + \sum_{i \in M} [R_i \cdot \ln(A_i) - r_i \cdot \ln(\alpha_i)] + \sum_{j \in N} [C_j \cdot \ln(B_j) - c_j \cdot \ln(\beta_j)] + \sum_{(i,j) \in S} \sum_{l \in H} \ln(\gamma_{ijl})$$

subject to the constraints

$$(10.25) \qquad [-\ln(\delta) - \ln(\alpha_i) + \ln(A_i) - \ln(\beta_j) + \ln(B_j) + \ln(\gamma_{ijl})] \geq \ln(\tfrac{p_{ij}}{d_l}) \quad \forall\, (i,j) \in S, l$$

$$(10.26) \qquad \delta > 0,\ \alpha_i \geq 1,\ A_i \geq 1,\ \beta_j \geq 1,\ B_j \geq 1,\ \gamma_{ijl} \geq 1$$

We isolate $\ln(\gamma_{ijl})$ on the left hand side of (10.25) and take the exponent on both sides, which result in the equivalent constraint:

$$(10.27) \qquad \gamma_{ijl} \geq \frac{\delta \cdot \frac{\alpha_i \cdot \beta_j \cdot p_{ij}}{A_i \cdot B_j}}{d_l} \qquad \forall\, (i,j) \in S, l$$

By introducing the variables $\lambda_i = \frac{\alpha_i}{A_i}$ and $\mu_j = \frac{\beta_j}{B_j}$, (10.27) is simplified to:

$$(10.28) \qquad \gamma_{ijl} \geq \frac{\delta \cdot \lambda_i \cdot \mu_j \cdot p_{ij}}{d_l} \qquad \forall\, (i,j) \in S, l$$

Regarding the new variables, we deduce from (10.26) that:

(10.29) $\lambda_i > 0$ and $\mu_j > 0$

We use the complementary slackness conditions to say something about the values of the dual variables. These conditions tell us the following: If a constraint in the primal problem is not binding, then the dual variable associated with this constraint is equal to zero. Moreover, a binding constraint in the primal problem has an associated dual variable which is greater than or equal to zero. Below we investigate the constraints in the primal problem.

When $\ln(\alpha_i) > 0$, i.e. $\alpha_i > 1$, constraint (10.16) is binding, which means that the number of seats given to constituency i is at the lower bound r_i. Similarly, when $A_i > 1$, (10.17) is a binding constraint and the number of seats given to constituency i is at the upper bound R_i Obviously, (10.16) and (10.17) cannot be binding at the same time. The described relationships have the following implications for the variable λ_i:

(10.30) $\lambda_i > 1$ implies $a_{iNH} = r_i$ and $\lambda_i < 1$ implies $a_{iNH} = R_i$

The reasoning for the dual variables associated with constraints (10.18) and (10.19) is similar, with the following implications for μ_j:

(10.31) $\mu_j > 1$ implies $a_{MjH} = c_j$ and $\mu_j < 1$ implies $a_{MjH} = C_j$

Notice the similarity between (10.29) - (10.31) and Definition 9.7.

Constraint (10.23) is binding when cell (i,j) has got its lth seat. Then the value of the associated dual variable is $\ln(\gamma_{ijl}) \geq 0$, i.e. $\gamma_{ijl} \geq 1$. Moreover, if $l \geq 2$ in this situation, it follows from the construction of the variable a_{ijl} that $\gamma_{ij(l-1)} > \gamma_{ijl} \geq 1$. The following relationship exists between γ_{ijl} and a_{ijl}:

(10.32) $\gamma_{ijl} > 1$ implies $a_{ijl} = 1$

The equality constraint for the total number of seats (10.15) can be written as two inequality constraints, one with h as the upper bound and the other with h as the lower bound. Thus, the dual variable δ is of the same type as λ_i and μ_j. $\delta < 1$ implies that the upper bound is binding, while $\delta > 1$ implies that the lower bound is binding.

10.3 Utility approach

Let us generalize the utility approach from section 6.9. In the matrix apportionment context we maximize the utility over all cells:

$$\max \sum_{i \in M} \sum_{j \in N} [p_{ij} \cdot u_{ij}(l)] = \max \sum_{(i,j) \in S} \sum_{l \in H} [(\frac{p_{ij}}{d_l}) \cdot a_{ijl}] \tag{10.33}$$

subject to the same constraints as for the entropy minimization problem. Because $p_{ij} \cdot a_{ijl} = 0$ when $(i,j) \in \bar{S}$, it would not have made any difference if we had summed over all (i,j) in (10.33).

Let us formulate the dual to this utility maximization problem. In this formulation we let the dual variables associated with the different constraints be: δ for (10.15), α_i for (10.16), A_i for (10.17), β_j for (10.18), B_j for (10.19), and γ_{ijl} for (10.23). The result is the following the dual problem:

$$\begin{aligned} \min \quad & -h \cdot \delta + \sum_{i \in M} (R_i \cdot A_i - r_i \cdot \alpha_i) \\ & + \sum_{j \in N} (C_j \cdot B_j - c_j \cdot \beta_j) + \sum_{(i,j) \in S} \sum_{l \in H} \gamma_{ijl} \end{aligned} \tag{10.34}$$

subject to the constraints

$$\gamma_{ijl} \geq \frac{p_{ij}}{d_l} + \delta + (\alpha_i - A_i) + (\beta_j - B_j) \qquad \forall (i,j) \in S, l \tag{10.35}$$

$$\delta \text{ unconstrained}, \alpha_i \geq 0, A_i \geq 0, \beta_j \geq 0, B_j \geq 0, \gamma_{ijl} \geq 0 \tag{10.36}$$

We now introduce the variables λ_i and μ_j, which here are defined as $\lambda_i = \alpha_i - A_i$ and $\mu_j = \beta_j - B_j$ respectively. Both λ_i and μ_j are unconstrained in sign. The new variables simplify (10.35) to:

$$\gamma_{ijl} \geq \frac{p_{ij}}{d_l} + \delta + \lambda_i + \mu_j \qquad \forall\ (i,j) \in S, l \tag{10.37}$$

A similar analysis to that in section 10.2 can be carried out for the dual variables. The results are essentially the same when differences in definitions of dual variables are taken into account.

We round off this chapter with a comparison of the utility maximization formulation and the entropy minimization formulation. What is different in these two formulations are the objective functions. However, given the set of possible apportionments $R^0(\mathbf{p},\sigma)$, which is equal for both formulations, the two objective functions (10.14) and (10.33) result in identical optimal apportionments. Here follows a brief explanation: Let (10.33) be the objective function and order the quotients $\frac{p_{ij}}{d_l}$ in descending order. This ordering will not be altered by making a monotonic transformation, such as taking the natural logarithm $\ln(\frac{p_{ij}}{d_l})$ or the square root $\sqrt{\frac{p_{ij}}{d_l}}$. Even though (10.33) and (10.14) are different, the shaping of the convex set $R^0(\mathbf{p},\sigma)$ ensures that both objective functions attain their highest value for the same extreme point(s) of $R^0(\mathbf{p},\sigma)$.

We choose to work with the logarithmic version of the objective function, i.e. (10.14), in the remainder of this dissertation. This means multiplicative scaling to find the optimal apportionment. Use of (10.33) would have meant additive scaling, compare (10.37) with (10.28).

Chapter 11: Relaxed formulations

The matrix apportionment problem as formulated at the end of section 10.1 is a linear programming problem which can be solved by the simplex method. Because of the problem's simple structure, our hope was to find a simpler algorithm. Another way of attacking the problem is to relax one set of constraints like Helgason and Jörnsten do. Their formulation, which forms the basis for later algorithms, is presented in the first section. In section 11.2 we introduce a measure for evaluating the error of the current assignment. A good starting point is important for every algorithm, and in the third section we come up with three initialization procedures for the multipliers. We carry out a test to determine a good parameter value for the most advanced of these procedures in section 11.5. Section 11.4 considers implications of relaxing the utility maximization formulation. The final section is devoted to a discussion about politically acceptable algorithms for the matrix apportionment problem.

11.1 Helgason and Jörnsten's formulation

To solve the matrix apportionment problem, [H&J] (page 21) relax the lower (10.16) and upper bound (10.17) constituency constraints. This leads to the following relaxed formulation, where $\ln(\alpha_i)$ and $\ln(A_i)$ are the dual variables for the lower and upper bound constraint respectively:

$$(11.1) \qquad \min_{\alpha \geq 1} \min_{A \geq 1} L(\alpha, A)$$

where the function $L(\alpha,A)$ is defined as:

$$(11.2) \qquad L(\alpha,A) = \sum_{i \in M} [R_i \cdot \ln(A_i) - r_i \cdot \ln(\alpha_i)] + \max_{\mathbf{a}} \sum_{(i,j) \in S} \sum_{l \in H} [\ln(\frac{\alpha_i \cdot p_{ij}}{A_i \cdot d_l}) \cdot a_{ijl}]$$

subject to the constraints

$$(11.3) \qquad \sum_{i \in M} \sum_{j \in N} \sum_{l \in H} a_{ijl} = a_{MNH} = h$$

$$(11.4) \qquad \sum_{i \in M} \sum_{l \in H} a_{ijl} = a_{MjH} \geq c_j \qquad \forall j$$

$$(11.5) \qquad \sum_{i \in M} \sum_{l \in H} a_{ijl} = a_{MjH} \leq C_j \qquad \forall j$$

$$(11.6) \qquad a_{ijl} \geq 0 \qquad \forall (i,j) \in S, l$$

$$(11.7) \qquad a_{ijl} \leq 1 \qquad \forall (i,j) \in S, l$$

$$(11.8) \qquad a_{ijl} = 0 \qquad \forall (i,j) \in \bar{S}, l$$

The problem is a non-linear minimization problem. It can be solved by a subgradient method as described in [H&J] (page 22-23).

$L(\alpha,A)$ is an auxiliary tool in the search for the optimal apportionment. We evaluate $L(\alpha,A)$ by using a greedy algorithm. This algorithm assigns seats to the best adjusted quotients $\ln(\frac{\alpha_i \cdot p_{ij}}{A_i \cdot d_l})$. The assignment process works as follows: First a number of seats corresponding to the lower party bound c_j is given out within each party. When the problem is equality constrained, this is the end of the assignment process. For problems with inequality party constraints there are still $h - c_N$ seats left for distribution. These seats are given out globally, always respecting the parties' upper bounds **C**, until all h seats have been distributed.

Rather than relaxing the constituency constraints, we could relax the party constraints. With this relaxed formulation, the objective would have been to minimize the function L(β,B):

$$(11.9) \qquad L(\beta,B) = \sum_{j \in N} [C_j \cdot \ln(B_j) - c_j \cdot \ln(\beta_j)] + \max_{\mathbf{a}} \sum_{(i,j) \in S} \sum_{l \in H} [\ln(\frac{\beta_j \cdot p_{ij}}{B_j \cdot d_l}) \cdot a_{ijl}]$$

subject to constraints (11.3) - (11.8), but with (10.16) - (10.17) substituted for (11.4) - (11.5).

In the following we refer to the formulation with relaxed constituency constraints as **constituency relaxation**. Similarly, the formulation with relaxed party constraints is referred to as **party relaxation**. We have included both relaxations in the tests later in the dissertation. The description below and in chapter 12 is based on constituency relaxation.

11.2 The measure of goodness

Let $\tilde{\mathbf{a}}$ denote the current assignment. To check whether this assignment is feasible, we compare it with the constituency constraints. If it respects all of them and the variables $\lambda_i = \frac{\alpha_i}{A_i}$ satisfy condition (10.30), an optimal apportionment has been found. Until this happens the actual assignment will violate one or more of the relaxed constituency constraints. We need a measure which evaluates the magnitude of these violations and define the **measure of godness**, denoted ρ, for this purpose:

$$(11.10) \qquad \rho = \sum_{i \in M^-} (r_i - \tilde{a}_{iNH}) + \sum_{i \in M^+} (\tilde{a}_{iNH} - R_i)$$

where M^- is the set of constituencies which currently are assigned less seats than their lower bound, while M^+ is the set of constituencies with more seats than

their upper bound. The set M^0 consists of the remaining constituencies, i.e. the constituencies for which the current assignment does not violate any bound. Let us formulate these sets mathematically:

(11.11)
$$M^- = \{i \in M: \ \tilde{a}_{iNH} < r_i\}$$
$$M^0 = \{i \in M: \ r_i \le \tilde{a}_{iNH} \le R_i\}$$
$$M^+ = \{i \in M: \ \tilde{a}_{iNH} > R_i\}$$

The constituencies in M^-, M^0, and M^+ are called **underrepresented**, **rightly represented**, and **overrepresented** respectively. A constituency which is not rightly represented, i.e. is a member of either M^- or M^+, is sometimes called **malrepresented.**

Let us go back and take a closer look at (11.10). From the integer requirement regarding a_{ijl}, the integer assumption regarding r_i and R_i, and the definition of the sets M^- and M^+, it is clear that ρ is integer valued and non-negative. This is natural since ρ shows the violation of the constituency constraints measured in number of seats. ρ can be viewed as the total error of the current assignment. The ultimate goal is to find an assignment for which $\rho = 0$.

$L(\alpha,A)$ is a continuous function of $\alpha > 0$ and $A > 0$. It is differentiable everywhere with the exception of tie points in the assignment. The subgradients of $L(\alpha,A)$ with respect to α_i and A_i are:

(11.12)
$$\frac{\partial L(\alpha,A)}{\partial \alpha_i} = \frac{1}{\alpha_i} \cdot \Big(\sum_{j \in N} \sum_{l \in H} a_{ijl} - r_i\Big) = \frac{1}{\alpha_i} \cdot (a_{iNH} - r_i)$$

(11.13)
$$\frac{\partial L(\alpha,A)}{\partial A_i} = \frac{1}{A_i} \cdot \Big(R_i - \sum_{j \in N} \sum_{l \in H} a_{ijl}\Big) = \frac{1}{A_i} \cdot (R_i - a_{iNH})$$

In the derivation of these subgradients we have utilized that $a_{ijH} = 0$ for $(i,j) \in \bar{S}$ means that $\sum_{(i,j) \in S} a_{ijH}$ can be expressed as $\sum_{i \in M} \sum_{j \in N} a_{ijH}$.

By studying (11.12) - (11.13), we see that the subgradients are equal to zero when the assigned number of seats is equal to r_i and R_i respectively. $\frac{\partial L(\alpha,A)}{\partial \alpha_i}$ is

positive (negative) when constituency i has been assigned more (less) seats than the lower bound r_i, while $\frac{\partial L(\alpha,A)}{\partial A_i}$ is negative (positive) when the assignment gives constituency i more (less) seats than the upper bound R_i. Obviously, both subgradients cannot be positive or negative at the same time. The subgradients are of interest because they tell us in which direction α and A shall be adjusted to reduce ρ and attain a lower value of $L(\alpha,A)$.

We move on to the equality constrained problem, i.e. $r_i = R_i$ etc. Then the measure of goodness can be written as:

$$\rho = \sum_{i \in M} |\tilde{a}_{iNH} - R_i| \tag{11.14}$$

ρ is now a non-negative even number. Let us explain why: Because $r_M = R_M = h$ in the equality constrained problem, underrepresentation for one constituency implies that there are one or more other constituencies with offsetting overrepresentation. Since every seat which violates a constituency constraint thereby is counted twice, ρ is an even number.

$r_i = R_i$ and $c_j = C_j$ in the equality constrained problem, so we let R_i and C_j denote the bounds in the following. By making use of $\lambda_i = \frac{\alpha_i}{A_i}$, the relaxed formulation from section 11.1 can be simplified to:

$$\begin{aligned} &\min \quad L(\lambda) \\ &\lambda > 0 \end{aligned} \tag{11.15}$$

$$L(\lambda) = \sum_{i \in M} [-R_i \cdot \ln(\lambda_i)] + \max_{\mathbf{a}} \sum_{(i,j) \in S} \sum_{l \in H} [\ln(\frac{\lambda_i \cdot p_{ij}}{d_l}) \cdot a_{ijl}] \tag{11.16}$$

subject to the constraints

$$a_{MNH} = h \tag{11.17}$$

$$a_{MjH} = C_j \qquad \forall j \tag{11.18}$$

$$a_{ijl} \geq 0 \qquad \forall (i,j) \in S, l \tag{11.19}$$

$$a_{ijl} \leq 1 \qquad \forall (i,j) \in S, l \tag{11.20}$$

$$a_{ijl} = 0 \qquad \forall (i,j) \in \bar{S}, l \tag{11.21}$$

The subgradient of $L(\lambda)$ with respect to λ_i is:

$$\frac{\partial L(\lambda)}{\partial \lambda_i} = \frac{1}{\lambda_i} \cdot \left(\sum_{j \in N} \sum_{l \in H} a_{ijl} - R_i\right) = \frac{1}{\lambda_i} \cdot (a_{iNH} - R_i) \tag{11.22}$$

Thus, $\frac{\partial L(\lambda)}{\partial \lambda_i}$ is positive for an overrepresented, zero for a rightly represented, and negative for an underrepresented constituency. From now on we use the name **constituency multiplier** for λ_i. The optimal apportionment for an equality constrained problem has been found if $\rho = 0$. With $\mu_j = 1$ for all j and an appropriate value for δ, the constituency multiplier conditions (10.30) will be satisfied in such a situation.

11.3 Initialization of constituency multipliers

As the empirical test in chapter 14 will reveal, the number of iterations to solve a matrix apportionment problem depends on how good the initial multiplier set is. In the case of constituency relaxation, it is the initial constituency multipliers which are of interest. We denote the initial multiplier for constituency i by $\mathring{\lambda}_i$. Below we present three initialization procedures. These procedures are created for equality constrained problems, but with some minor modifications they are also applicable to inequality constrained problems.

Our review starts with the procedure called **no initialization**. For the constituency multipliers to have no effect on the quotients, i.e. $\ln(\mathring{\lambda}_i \cdot \frac{p_{ij}}{d_j}) = \ln(\frac{p_{ij}}{d_j})$, we let $\mathring{\lambda}_i = 1$ for all i. The work at the initialization stage is minimized by this approach.

No initialization does not utilize any of the available data. With **quota ratio initialization** we take the constituency bounds R_i and the number of votes in each constituency p_{iN} into consideration. The average number of people per seat for a constituency $p_i = \frac{p_{iN}}{R_i}$ usually deviates from the corresponding national average $p = \frac{p_{MN}}{h}$. To make the adjusted average number of people per seat $\mathring{\lambda}_i \cdot p_i$

equal for all constituencies, we determine $\mathring{\lambda}_i$ from the following ratio:

(11.23) $$\mathring{\lambda}_i = \frac{p}{p_i} = \frac{R_i}{q_i} \qquad \text{for all } i$$

$\mathring{\lambda}_i < 1$ if constituency i's quota has been rounded down to R_i, while $\mathring{\lambda}_i > 1$ if the quota has been rounded up to R_i. Notice that $(\mathring{\lambda}_i - 1)$ shows the extra influence of the average voter from constituency i compared with the average national voter. Example 11.1 illustrates quota ratio initialization:

Example 11.1

Our data are taken from the Icelandic election in 1995. In this election there were 6 parties which won seats in the Althing. The total number of votes cast for these parties was $p_{MN} = 161948$. With a total of $h = 63$ seats, this means an average of $p = \frac{161948}{63} \approx 2571$ votes per seat. Austurland, which is one of the small constituencies, contributed to p_{MN} with $p_{Aul.N} = 7818$ votes. The Icelandic election law favours small constituencies. Austurland was secured a representation of 5 seats although its quota only was $q_{Aul} = \frac{7818}{161948} \cdot 63 \approx 3{,}04$ seats. The average number of votes behind each representative from Austurland was therefore $p_{Aul} = \frac{7818}{5} \approx 1564$. Formula (11.23) then gives an initial constituency multiplier for Austurland of $\mathring{\lambda}_{Aul} = \frac{2571}{1564} \approx 1{,}644$. Hence, the average voter in Austurland was 64,4% more influential than the average Icelandic voter in this election.

Quota ratio initialization only adjusts for the bias inherent in the constituency bounds. We therefore expect multiplier sets created by this initialization procedure to be a better starting point for unbiased than for biased divisor methods.

The last procedure we consider is called **apportionment initialization**. In addition to the constituency bounds, this procedure utilizes the whole vote matrix plus the divisor method which is going to be used for the matrix apportionment. For each constituency we combine votes and constituency bound R_i into an election situation. Thereafter, we make the vector apportionment with the actual divisor method, i.e. we rank the quotients $\frac{p_{ij}}{d_l}$ within the constituency. The reason

for carrying out this apportionment is to identify the quotient which qualifies for the last seat q_{R_i} and the best quotient which does not qualify for a seat $q_{(R_i+1)}$. We call the weighted average of these two quotients **target quotient** and denote it t_i:

(11.24) $$t_i = \tau \cdot q_{R_i} + (1 - \tau) \cdot q_{(R_i+1)} \qquad \text{where } 0 \le \tau \le 1 \text{ and real}$$

The target quotient is a sort of marginal value of representation for the constituency. It depends on the choice of **target weight** τ and the divisor method applied. In the penultimate section we carry out a test to find a good τ value for our further investigations.

We use the logarithmic version of the target quotient, i.e. $\ln(t_i)$, along with the relaxed formulation from section 11.1. To even out representational differences between constituencies, we require that all multiplier adjusted target quotients shall be equal. This is achieved by setting the value of every multiplier adjusted target quotient equal to the **initial marginal value of representation**, denoted $\mathring{\kappa}$. Thus:

(11.25) $$\ln(\mathring{\lambda}_i \cdot t_i) = \ln(\mathring{\kappa}) \quad \Leftrightarrow \quad \ln(\mathring{\lambda}_i) = \ln(\mathring{\kappa}) - \ln(t_i)$$

which implies that the initial multiplier for constituency i is given as:

(11.26) $$\mathring{\lambda}_i = \mathring{\kappa} \cdot \frac{1}{t_i}$$

We have not said anything about the value of $\mathring{\kappa}$ yet. Before we do so notice that a logarithmic quotient can be written as: $\ln(\mathring{\lambda}_i \cdot \frac{p_{ij}}{d_l}) = \ln(\mathring{\lambda}_i) + \ln(\frac{p_{ij}}{d_l})$. From (11.26) we see that all initial constituency multipliers are scaled by the same factor by a change in $\mathring{\kappa}$. Since all quotients within a constituency are adjusted by the same amount via the constituency multiplier, the **initial measure of goodness**, denoted $\mathring{\rho}$, is independent of $\mathring{\kappa}$. The value of $\mathring{\kappa} > 0$ can therefore be chosen arbitrarily. Our choice is:

(11.27) $$\mathring{\kappa} = \frac{p_{MN}}{h + (1 - \tau)\cdot m}$$

Notice that (11.27) takes the effect of τ into consideration via the denominator. For $\tau = 1$, $\mathring{\kappa}$ is the average number of votes per seat. If $q_{(R_i + 1)}$ is given all the weight in (11.24), i.e. $\tau = 0$, $\mathring{\kappa} = \frac{1}{h + m}$. We illustrate apportionment initialization with the following numerical example:

Example 11.2

We continue to utilize data from the Icelandic election in 1995 and the constituency Austurland in particular. Iceland is divided in 8 constituencies. With the target weight set to $\tau = 0{,}3$, (11.27) gives the following initial marginal value of representation: $\mathring{\kappa} = \frac{161948}{63 + (1 - 0{,}3)\cdot 8} \approx 2361$. To carry out the apportionment initialization we need to know the parties' votes in Austurland and which divisor method to apply. Let us apply HA, which results in the following quotients:

Table 11.1

Party	A	B	D	G
1st quotient	$\frac{577}{1} = 577$	$\frac{3668}{1} = 3668$ I	$\frac{1760}{1} = 1760$ III	$\frac{1257}{1} = 1257$ IV
2nd quotient		$\frac{3668}{2} = 1834$ II	$\frac{1760}{2} = 880$	$\frac{1257}{2} \approx 629$
3rd quotient		$\frac{3668}{3} \approx 1223$ V		
4th quotient		$\frac{3668}{4} = 917$ VI		

Ranking of quotients in Austurland with HA as apportionment method.

Parties with less support than A have been omitted from Table 11.1. The roman numerals show the ranking of the six largest quotients. Since Austurland sends five representatives to the Althing, it is the fifth and sixth quotient which are of interest. We put their values into (11.24) to determine Austurland's target quotient: $t_{Aul} = 0{,}3 \cdot 1223 + 0{,}7 \cdot 917 \approx 1009$. Finally, the initial constituency multiplier for Austurland is found from (11.26): $\mathring{\lambda}_{Aul} = \frac{2361}{1009} \approx 2{,}340$.

The advantage of apportionment initialization over quota ratio initialization is its relation to the local characteristics within each constituency. On the other hand, apportionment initialization requires more work. While quota ratio initialization is independent of the divisor method being applied, each divisor method results in its own initial muliplier set with apportionment initialization. We expect multiplier sets determined by apportionment initialization to be a good starting point for most divisor methods.

11.4 Consequences for the additive version

In this section we look at the consequences of relaxing the constituency constraints in the utility maximization formulation from section 10.3. With α and A as the dual variables associated with the constituency constraints, the relaxed formulation becomes:

$$\min_{\alpha \geq 0} \min_{A \geq 0} L(\alpha, A) \tag{11.28}$$

$$L(\alpha,A) = \sum_{i \in M} [R_i \cdot A_i - r_i \cdot \alpha_i] + \max_{\mathbf{a}} \sum_{(i,j) \in S} \sum_{l \in H} [(\frac{p_{ij}}{d_l} + \alpha_i - A_i) \cdot a_{ijl}] \tag{11.29}$$

subject to constraints (11.3) - (11.8).

For the equality constrained problem the adjusted quotients are $\frac{p_{ij}}{d_l} + \lambda_i$, where $\mathring{\lambda}_i$ can be interpreted as a constituency handicap. Moreover, the subgradient of $L(\lambda)$ with respect to λ_i is $\frac{\partial L(\lambda)}{\partial \lambda_i} = a_{iNH} - R_i$.

The initialization procedures must be modified to suit the additive version. They are now carried out as follows: With no initialization we set $\mathring{\lambda}_i = 0$ for all i. Each constituency's initial handicap is determined as $\mathring{\lambda}_i = p - p_i$ with quota ratio initialization. Regarding apportionment initialization, the target quotients are calculated by (11.24) as before. However, the formula for $\mathring{\lambda}_i$ has changed:

(11.30) $\quad t_i + \mathring{\lambda}_i = \mathring{\kappa} \quad \Leftrightarrow \quad \mathring{\lambda}_i = \mathring{\kappa} - t_i$

Like for the logarithmic version, the initial measure of goodness $\mathring{\rho}$ is independent of the value choosen for $\mathring{\kappa}$. We therefore retain our choice from (11.27). However, for a given election situation, the value of $\mathring{\rho}$ with the additive version is not generally the same as with the logarithmic version.

11.5 Determination of a good value for τ

The target weight τ used for apportionment initialization influences the initial multiplier set in such a way that the initial measure of goodness $\mathring{\rho}$ depends on the chosen τ value. Since we expect the number of iterations to find the optimal apportionment to be positively correlated with $\mathring{\rho}$, we want to identify a τ value which makes the average $\mathring{\rho}$ "small". For this purpose we have carry out a test:

In the test we utilize data from general elections in six countries. These data sets are edited the following way: We omit all parties with less than 1% support. Finland and Denmark have some constituencies with special status. We omit Åland from the Finnish set and the Faroe Islands and Greenland from the Danish set.

For a given vote matrix, there are two main features which distinguish a matrix problem from another, namely the bound vector σ and the matrix apportionment method. In this test we formulate all problems as equality constrained matrix apportionment problems. We operate with 5 different bound vectors for each country; the real election bounds plus 4 bound vectors determined by vector apportionments based on the vote totals. We use SD, DM, MF, and HA for these vector apportionments and determine both constituency **R** and party bounds **C** by the same apportionment method. For each bound vector we operate with 4 different divisor methods; $CP_{0,01}$, DM, MF, and HA. Thus, for every country

(vote matrix) there are $5 \cdot 4 = 20$ combinations of bound vector and divisor method. Let us call each such combination **country set**.

$\mathring{\rho}$ is not a monotonic function of τ over the interval $0 \leq \tau \leq 1$. This means that the determination of the optimal target weight(s) would require a lot of work. Our aim is only to find a good target weight, so we limit the test to 11 τ values, ranging from $\tau = 1$ down to $\tau = 0$. For each combination of target weight and country set we calculate target quotients and determine initial constituency multipliers. There are a total of $11 \cdot 20 = 220$, possibly different, multiplier sets for each country. For each multiplier set we make the assignment as described in section 11.1 and calculate $\mathring{\rho}$ by (11.14).

Table 11.2 summarizes the results of the test. The first row shows the data sets utilized, while the bottom part of the table gives information about the number of constituencies and parties involved. Each entry in the main body of the table is the average $\mathring{\rho}$ for the particular combination of τ and country set, i.e. the average over 20 cases. The last column contains the sum of averages for each τ value.

Table 11.2

τ	ICE 91	AUT 90	FIN 91	NOR 93	SWE 91	DEN 84	Sum
1,0	4,2	4,9	12,0	17,4	19,0	23,3	80,8
0,9	2,7	5,2	11,5	15,9	17,2	22,8	75,3
0,8	2,5	5,2	11,4	15,8	17,0	22,8	74,7
0,7	**2,2**	5,1	11,6	15,6	16,7	22,7	73,9
0,6	**2,2**	5,1	11,6	15,3	16,7	22,5	73,4
0,5	**2,2**	5,0	11,4	15,2	16,2	22,5	72,5
0,4	2,5	4,7	11,2	15,2	**15,9**	22,5	72,0
0,3	2,5	4,7	11,0	14,7	**15,9**	22,5	**71,3**
0,2	2,5	4,5	11,1	14,7	16,0	22,5	**71,3**
0,1	2,7	**4,3**	11,3	**14,3**	16,9	22,4	71,9
0,0	3,3	4,4	**10,9**	14,7	17,8	**22,1**	73,2
m	8	9	14	19	28	17	
n	5	5	9	8	8	10	

Average initial measure of goodness $\mathring{\rho}$ with constituency relaxation.

As one could expect, the data sets result in different $\mathring{\rho}$ values. $\mathring{\rho}$ is lowest for Iceland and highest for Denmark. Among many causes $\mathring{\rho}$ seems to depend on the number of constituencies *m* and parties *n*. The lowest average for each country is

in bold print. We observe that the best target weight differs from one country to the next. Based on the sum column, target weights in the region $0,1 \leq \tau \leq 0,4$ seem preferable, i.e. the best quotient which does not qualify for a seat $q_{(R_i + 1)}$ should count most in the calculation of the target quotient.

As shown at the end of section 11.1, it is possible to relax the party constraints instead of the constituency constraints. Since we deal with equality constrained problems, we introduce $\mu_j = \frac{\beta_j}{B_j}$, which simplifies $L(\beta,B)$ to $L(\mu)$. This results in a formulation analogous to that given by (11.15) - (11.21). We carry out the target weight test for this formulation too. To utilize our existing program for constituency relaxation, we transpose the election data such that constituencies become "parties" and vice versa. Table 11.3 presents the results of the target weight test baseed on party relaxation:

Table 11.3

τ	AUT 90	FIN 91	ICE 91	DEN 84	NOR 93	SWE 91	Sum
1,0	3,4	7,2	7,2	7,3	7,7	9,1	41,9
0,9	3,4	5,3	6,7	5,5	7,3	8,6	36,8
0,8	3,0	5,3	6,7	**5,3**	**7,2**	8,5	36,0
0,7	3,0	4,9	6,6	**5,3**	**7,2**	8,3	**35,3**
0,6	2,9	**4,7**	6,6	5,6	7,5	8,3	35,6
0,5	**2,6**	4,9	6,5	5,4	7,7	**8,2**	**35,3**
0,4	2,9	5,1	6,0	5,6	7,9	8,4	35,9
0,3	2,9	5,1	5,7	5,6	8,0	8,9	36,2
0,2	2,8	5,3	5,7	5,5	8,4	8,9	36,6
0,1	3,4	5,5	**5,2**	6,0	8,6	9,0	37,7
0,0	4,1	6,5	5,4	6,6	8,9	9,0	40,5

Average initial measure of goodness $\mathring{\rho}$ with party relaxation.

Comparison of Table 11.2 and Table 11.3 reveals that $\mathring{\rho}$ is lower for all countries but Iceland with party relaxation. This has consequences for the sums which are about 50% lower in the latter table. The target weights $\tau = 0,5$ and $\tau = 0,7$ have the lowest sums in Table 11.3. However, all τ values in the region $0,2 \leq \tau \leq 0,9$ seem acceptable. Most τ values seem acceptable when the two tables are considered together. We choose to work with $\tau = 0,3$ in the following because this value gives the lowest sum of sums. However, there is little difference between target weights in the region $0,2 \leq \tau \leq 0,5$ based on this criterion.

It is worth noticing that the end points of the target weight interval [0, 1] often fare badly in comparison with other target weights. A possible explanation is that the target quotient only represents one quotient when $\tau = 1$ or $\tau = 0$, while it is a weighted average of two quotients when $0 < \tau < 1$. In other words; two quotients tell more about the competition for the last seat than one quotient alone is able to do.

11.6 Politically acceptable algorithms

Both the algorithm described in [H&J] (page 23) and our multiplier adjustment algorithm presented in the next chapter are iterative. Be aware that any exact solution of the matrix apportionment problem would require an iterative process. Based on discussions with politicians Helgason & Jörnsten have reason to believe that iterative algorithms never would be politically acceptable. What characterizes an iterative algorithm is that the assignment changes from one iteration to the next. It is this kind of behaviour, where a quotient which already has been assigned a seat might lose it at a later stage of the algorithm, the politicians see as unacceptable. We let this view be the premise for the further discussion in this section.

In oral communication Helgason & Jörnsten have stated that their main reason for studying an assignment process of the kind presented in the first section was to discover a good approximative solution method which might be politically acceptable in practical legal implementation. They presume that an algorithm has to be recursive to be acceptable, [H&J] (page 20). This point of view combined with their assignment process mean that the problem of creating an algorithm boils down to constructing a good initialization procedure.

In Helgason & Jörnsten's formulation from section 11.1 the constituency constraints are relaxed. This means initialization of constituency multipliers. An interesting question is whether an approximative algorithm based on party

relaxation is acceptable. [H&J] (page 21) do not believe so. The reason is as follows: Party relaxation would mean initialization of party multipliers. These multipliers represent a sort of valuation of the parties. This is presumably more controversial politically than a valuation of the constituencies. Many countries have some intended favouring of special types of constituencies, so one can argue that there already exists some kind of valuation of the constituencies.

As concluded above, our task is to construct an initialization procedure which is able to provide good multiplier sets. It is natural to start by taking a closer look at the performance of the three initialization procedures presented in section 11.3. In chapter 14 we carry out a test which reveals that apportionment initialization results in the lowest average $\mathring{\rho}$ in almost all situations. (Data which show the average $\mathring{\rho}$ for individual countries can be found in Appendix 3.) Below we explain how apportionment initialization can be tailor-made for a specific country.

In the previous section we saw that $\tau = 0{,}3$ was the best target weight for a mix of many different election situations. The election situations within a single country are much more stable, which simplifies the determination process for τ. In this case the determination process should be based on relevant historical data sets. By relevant we mean that the elections included shall have taken place with about the same rules and constituency structure as today. In situations where large changes in the election parameters m, h etc are planned, one should base the determination process on vote matrices representing plausible scenarios. Because the divisor method to apply for the matrix apportionment plus the way to determine the bound vector will be fixed by law, there is need for only one combination of bound vector and divisor method per data set in this determintion process. We have not determined an own τ for each country.

In search of better initialization procedures we look at possible modifications of apportionment initialization in this paragraph. The first suggestion is based on the observation that apportionment initialization gives poor initializations for particular groups of constituencies when HA and especially $CP_{0,01}$ are used as matrix

apportionment methods. A way of creating better initializations for these constituencies could be to adjust their multipliers found by apportionment initialization by a fixed amount $\bar{\kappa}$, i.e. let $\mathring{\lambda}_i = (\mathring{\kappa} \cdot \frac{1}{t_i}) + \bar{\kappa}$, where $\bar{\kappa} \in \langle -(\max_i (\mathring{\kappa} \cdot \frac{1}{t_i})), \infty \rangle$. All other constituency multipliers should still be determined by (11.26). Another idea is to base apportionment initialization on a specific divisor method, for example MF, irrespective of the divisor method which is going to be used for the matrix apportionment. Yet another idea is to combine aspects of different initialization procedures, e.g. by combining apportionment initialization with quota ratio initialization. Neither of the ideas presented above have been tried out.

Let us illustrate what a positive value of ρ might mean for an individual constituency when the problem is equality constrained: The most extreme consequence is that a constituency with $R_i = \frac{\rho}{2}$ is not awarded any seat. Based on the experience from the test in chapter 14 we can say that this outcome is not very likely. What is more likely is that the largest constituency is awarded $\frac{\rho}{2}$ seats too much. This illustration leads up to the question: How large might $\mathring{\rho}$ be without the initialization procedure being unacceptable? We relate the answer to the current situation regarding supplementary seats, see description in section 8.6. There is more freedom regarding the apportionment to individual constituencies with free supplementary seats, i.e. with a partial inequality constrained formulation, than with preassigned supplementary seats, i.e. with an equality constrained formulation. A consequence is that it is easier to achieve a low $\mathring{\rho}$ with free supplementary seats. As indirectly stated before, $\rho = 0$ cannot be guaranteed with a recursive algorithm. For countries with a relatively large number of free supplementary seats, like Denmark and Sweden, it is natural to require that $\mathring{\rho} = 0$ or very close to 0. In countries where all supplementary seats are preassigned or with no supplementary seats at all, like Iceland and Finland respectively, it is often difficult to achieve $\mathring{\rho} = 0$. It seems reasonable to allow a somewhat larger $\mathring{\rho}$ in these situations. The magnitude of m, n, etc should also be taken into consideration when deciding on the upper limit for $\mathring{\rho}$. A critical question in this connection is whether an approximative matrix apportionment would be better than the current approach with supplementary seats. We have not tested this matter.

In the beginning of this section we mentioned the point of view that iterative algorithms are unacceptable because of changing assignments during the apportionment process. Let us now review the apportionment process when apportionment initialization is applied and seats are apportioned with the assignment process described in section 11.1: Apportionment initialization determines initial constituency multipliers by assigning seats within each constituency and calculating target quotients. Given these multipliers, the seats are apportioned within the parties. The resulting apportionment is usually different from the earlier assignments within the constituencies. This is clearly in conflict with the standard laid down at the beginning of this section. Thus, apportionment initialization is ruled out. Quota ratio initialization, however, which usually results in a higher $\mathring{\rho}$, would be in accordance with the standard.

We find iterative algorithms acceptable. In our view, the politicians' objection is based on a technicality since for given $\mathbf{p}$, σ, and matrix apportionment method there is no uncertainty regarding what is the optimal apportionment(s). Furthermore, whatever iterative algorithm employed to determine the optimal apportionment(s), there is no need for making the temporary assignments along the way public. Finally, it should not be that difficult to formulate a law text which captures the essence of the matrix apportionment problem. A possibility is to state the election law in terms of the multiplier conditions given in Definition 9.7.

Chapter 12: An apportionment algorithm

The matrix apportionment problem can be solved in several ways: [B&D-2] (page 205→) present an algorithm which borrows ideas from the out-of-kilter algorithm, but conclude that this algorithm "leaves something to be desired as a method for finding solutions in practice". The linear network flow formulation given in section 10.1 opens for use of the simplex method as solution method. Moreover, this formulation can be converted to an assignment problem formulation. Then a specialized solution method for the assignment problem, like the Hungarian method, can be used. As mentioned in section 11.1, [H&J] (page 22-23) present a subgradient method which solves the relaxed formulation of the matrix apportionment problem. Our apportionment algorithm presented in this chapter is also based on this relaxed formulation. This algorithm finds the optimal apportionment for the equality constrained problem. With some modifications it can also handle the partial inequality constrained problem. The algorithm is illustrated by a numerical example which covers the entire next chapter. Finally, our Pascal program of the algorithm can be found in Appendix 2.

The structure of this chapter is as follows: We start with a sketch of the apportionment algorithm. Section 12.2 presents formulas for the multiplier adjustment of an underrepresented constituency, while the third section describes the necessary modifications regarding the adjustment of an overrepresented constituency. In section 12.4 we determine a good value for the parameter called adjustment weight. The topic in the penultimate section is how to foresee the change in ρ a multiplier adjustment will bring about. In the last section we describe some practical problems we encountered while working with the algorithm. NB. The description in this chapter is based on that the equality constrained problem is being solved. R_i and C_j are used as notation for constituency and party bounds respectively.

12.1 A sketch of the algorithm

We start this section with a presentation of multiplier sets and their connection with the optimal apportionment. The choice of multipliers has been successful if all constituency constraints are satisfied, i.e. $\rho = 0$. There are usually infinitely many combinations of multiplier values which result in $\rho = 0$, i.e. give the optimal apportionment. We call a set of multipliers which does so a **suitable multiplier set**. This definition does not include any explicit multiplier condition and is therefore somewhat different from the definition of a proper multiplier set in connection with Definition 9.7. However, a suitable set for the equality constrained problem can easily be converted to a proper set by appropriate scalings. Unfortunately, the initial multiplier set will seldom be suitable. Let the target weight test in section 11.5 serve as an illustration: In that test there were a total of 47 instances where $\rho = 0$ after the apportionment initialization. 39 of these occurred for Iceland with constituency relaxation, while the other 8 occurred for Austria; 2 with constituency relaxation and 6 with party relaxation. With a total of 220 multiplier sets for each combination of country set and relaxation, this implies initial solution percentages of about 18% for Iceland with $L(\lambda)$, 1% for Austria with $L(\lambda)$, 3% for Austria with $L(\mu)$, and 0% for the other countries.

When the current set of constituency multipliers results in an assignment which violates one or more constituency constraints, i.e. $\rho > 0$, we must find a new multiplier set. The new set is found by adjusting one or more of the constituency multipliers. A possible adjustment process is the Uzawa-type algorithm described in [H&J] (page 22-23). We have developed an algorithm which utilizes characteristics of the matrix apportionment problem. In this algorithm it is of uppermost importance that the new assignment is not worse than the current one measured by ρ. Moreover, we only adjust one constituency multiplier in each iteration. The reason for this choice is given in section 12.6. Here follows a description of the main features of our apportionment algorithm:

Algorithm 12.1

Step 1: Given $\rho > 0$, we identify all constituencies which violate a constituency constraint and classify them. They either belong to the group of underrepresented or the group of overrepresented constituencies, denoted M^- and M^+ respectively. We denote the number of seats constituency i is away from being rightly represented by ρ_i:

$$\rho_i = |a_{iNH} - R_i| \tag{12.1}$$

Clearly, $\sum_{i \in M} \rho_i = \rho$. Thus, ρ_i is constituency i's contribution to the measure of goodness.

Step 2: In this step we single out which constituency to adjust. An advanced selection procedure is described in section 12.5. The procedure described here is simpler. It selects the most malrepresented constituency:

$$\max_{i \in (M^- \cup M^+)} \rho_i \tag{12.2}$$

When there is a tie for this position between constituencies from both M^- and M^+, we look at the previous iteration. If it was an upadjustment, i.e. involved a constituency from M^-, we choose to deal with a constituency in M^+ now, and vice versa. By breaking ties between M^- and M^+ this way, we get a sort of alternation of up and downadjustments. In section 12.6 we explain why it is preferable to make both up and downadjustments. When there is a tie regarding criterion (12.2) within the chosen malrepresentation group, we select one of the eligible constituencies. This selection is directed by a rotation scheme for the constituencies. For details of this rotation scheme we refer to the Pascal program of the algorithm in Appendix 2. We call the selection procedure described in this step **representation selection.**

Step 3: The main task in this step is to determine how much the selected constituency multiplier λ_u shall be adjusted to make constituency u rightly represented, i.e. make $\rho_u = 0$. This topic is treated thoroughly in the following two sections.

The direction of the adjustment is found from the subgradient given by equation (11.22): $L(\lambda)$ is reduced if we adjust λ_u upwards when u is underrepresented and downwards when u is overrepresented. The adjustment of λ_u changes the multiplier set. With the new multiplier set we assign seats as described in section 11.1 and calculate ρ. If $\rho = 0$, the optimal apportionment has been found, otherwise we return to step 1.

We end this section with a brief explanation of why our algorithm will find the optimal apportionment: $L(\lambda)$ is a continuous convex function which attains its minimum value when all constituency constraints are satisfied. Like the subgradient method described in [H&J], our algorithm makes use of $L(\lambda)$'s subgradients to lead us in the direction of this minimum. However, unlike that method, our algorithm let the measure of goodness ρ govern the multiplier adjustments. These adjustments are conducted such that ρ never increases and usually decreases, as explained in the last part of section 12.2. When $\rho = 0$ we have found the optimal apportionment and reached the floor/minimum of $L(\lambda)$.

12.2 Upadjustment

Let us first introduce some notation: The lth logarithmic quotient in cell (i,j) is denoted $q_{ijl} = \ln(\lambda_i \cdot \frac{p_{ij}}{d_l})$. Q is the set of quotients which currently are assigned a seat, while $\overline{Q}$ is its complement. Thus, $\tilde{a}_{ijl} = 1 \Leftrightarrow q_{ijl} \in Q$ and $\tilde{a}_{ij(l+1)} = 0 \Leftrightarrow q_{ij(l+1)} \in \overline{Q}$. We call quotients in Q and $\overline{Q}$ assigned and unassigned respectively.

In the description of the adjustment processes we focus on constituency u and party v. We step into the apportionment algorithm after an assignment has been made with the assignment process described in section 11.1. This assignment gives w seats to cell (u,v), where w is a non-negative integer. Since the assignment process distributes seats greedily within each party, party v is always assigned C_v seats. There are three possibilities regarding the assignment to constituency u: $\tilde{a}_{uNH} > R_u$, $\tilde{a}_{uNH} = R_u$, and $\tilde{a}_{uNH} < R_u$. In this section we assume

that u is underrepresented by $\rho_u > 0$ seats:

(12.3) $\rho_u = R_u - \tilde{a}_{uNH}$ where $u \in M^-$

The constituency multiplier λ_u must be adjusted upwards for u to be assigned more seats. For constituency u to become rightly represented, ρ_u of its unassigned quotients must each win a seat. The task ahead is to determine which unassigned quotients that shall be assigned seats. In cell (u,v) the candidates are the largest unassigned quotients $q_{uva} \in \overline{Q}$. We call these quotients **challenging quotients** and denote them $q^C_{uv}(x)$, where x is a positive integer. The xth challenging quotient in cell (u,v) is the xth largest quotient which does not get a seat with the current assignment. Clearly, this is the $(w + x)$th largest quotient in cell (u,v):

(12.4) $q^C_{uv}(x) = q_{uv(w+x)}$ where $x \in \{1, ..., (\rho_u + 1)\}$

All challenging quotients are from the same constituency so their internal ordering within party v is constant. For a challenging quotient in cell (u,v) to win a seat, one of party v's assigned quotients in other cells must lose its seat. To help us in the following, we set up a list of the assigned quotients in descending order, excluding the w assigned quotients in cell (u,v). We call the x smallest quotients on this list **benchmark quotients** and denote them $q^B_{uv}(x)$. The smallest benchmark quotient $q^B_{uv}(1)$ is closest to losing its seat. Quotients belonging to constituency u are not benchmark quotients for u because a constituency cannot win a seat from itself. The maximal number of benchmark quotients for constituency u within party v is equal to the total assignment to v minus the seats assigned to cell (u,v), i.e. $C_v - w$. When party v is assigned seats in other constituencies than u i.e. $C_v - w > 0$, the determination of benchmark quotients can be described mathematically as follows: Starting with $x = 1$, the xth benchmark quotient, where $x \in \{1, ..., \min [(\rho_u + 1), (C_v - w)]\}$, is determined recursively as:

(12.5) $q^B_{uv}(x) = \min_{s \in \{M \setminus u\}} q_{sva}$ where $q_{sva} \in Q$

Once a quotient q_{sva} has been used, it is eliminated from the further determination process. To suit (12.5), consider it unassigned in the remainder of the process. We may need up to $(\rho_u + 1)$ benchmark quotients for constituency u within party v. In situations where $\rho_u + 1 > C_v - w \geq 0$ we define one artificial benchmark quotient within party v by letting $q^B_{uv}(C_v - w + 1) = 50$. We explain why artificial benchmark quotients are needed in the last section.

The first challenging quotient must surpass the first benchmark quotient to win a seat. When the second challenging quotient tries to get a seat, the actual benchmark has changed to the second benchmark quotient etc. We may need to determine up to $(\rho_u + 1)$ such pairs of benchmark and challenging quotients within party v. For each pair we calculate the distance $d_{uv}(x)$ between the two quotients. Thus, the xth smallest distance within party v is:

$$(12.6) \qquad d_{uv}(x) = q^B_{uv}(x) - q^C_{uv}(x) \qquad \text{for } x \in \{1, ..., \min[(C_v - w + 1), (\rho_u + 1)]\}$$

By definition $q^B_{uv}(x) \geq q^C_{uv}(x)$, so $d_{uv}(x)$ is non-negative. Since $q^B_{uv}(x + 1) \geq q^B_{uv}(x)$ and $q^C_{uv}(x + 1) < q^C_{uv}(x)$, the distances within party v form a strictly increasing sequence, i.e. $d_{uv}(x + 1) > d_{uv}(x)$.

In the description above we focused on party v. Distances regarding constituency u must also be calculated within all other parties. To distinguish distance numbers belonging to different parties from each other, it is sometimes convenient to denote distance/quotient no. x within party j as x_j. We use this notation some places in the following. The time has come to determine the $(\rho_u + 1)$ smallest distances over all parties. We call these distances **global distances**. Starting with $y = 1$ and continuing up to $y = (\rho_u + 1)$, the yth global distance for constituency u, denoted $d_u(y)$, is determined recursively as:

$$(12.7) \qquad d_u(y) = \min_{j \in N} d_{uj}(x_j)$$

As in (12.5), once a distance $d_{uj}(x_j)$ has been used, it is not eligible in the further determination process. Algorithm 12.2 in section 12.5 describes an efficient way of determining the global distances. Contrary to the relationship between distances within a single party, there might be ties between global distances, i.e. $d_u(y + 1) \geq d_u(y)$. Ties occur when distances from different parties are equal.

For constituency u to be assigned exactly ρ_u seats more, the quotients from u must be adjusted upwards by more than $d_u(\rho_u)$, but less than $d_u(\rho_u + 1)$. We let χ_u denote the adjustment distance for constituency u and define it as a weighted average of $d_u(\rho_u)$ and $d_u(\rho_u + 1)$:

(12.8) $$\chi_u = \upsilon \cdot d_u(\rho_u) + (1 - \upsilon) \cdot d_u(\rho_u + 1) \quad \text{where } 0 < \upsilon < 1 \text{ and real}$$

χ_u is non-negative, but only in extreme tie situations is it equal to zero. Situations with $\chi_u = 0$ without there being a tie in the apportionment have occurred during some computer runs, see section 12.6 for an explanation. We call υ in formula (12.8) **adjustment weight**. Let us look at the effect of the adjustment when $d_u(\rho_u + 1) > d_u(\rho_u)$: When υ is in the neighbourhood of 1, the ρ_uth challenging quotient will be adjusted upwards such that it barely beats the ρ_uth benchmark quotient. The other extremity, with υ close to 0, results in an adjusted $(\rho_u + 1)$th challenging quotient which is a little bit smaller than the $(\rho_u + 1)$th benchmark quotient. Should it happen that the global distances $d_u(\rho_u)$ and $d_u(\rho_u + 1)$ are equal, the ρ_uth and $(\rho_u + 1)$th challenging quotients will be adjusted such that they become equal to their respective benchmark quotients. Dependent on how the ties are broken, u wins $(\rho_u - 1)$, ρ_u, or $(\rho_u + 1)$ seats in this case.

To complete the description of the upadjustment process, we present formulas for updating the quotient and multiplier values. Let $\bar{q}_{ujl}$ be the value of an arbitrary quotient from constituency u prior to the adjustment and q_{ujl} its value afterwards. Then q_{ujl} can be expressed as:

(12.9) $$q_{ujl} = \bar{q}_{ujl} + \chi_u$$

Let $\bar{\lambda}_u$ denote the multiplier value before the adjustment. By making use of the relationship $q_{ijl} = \ln(\lambda_i \cdot \frac{p_{ij}}{d_l})$ we derive the formula for the updated constituency multiplier λ_u:

$$\ln(\lambda_u \cdot \frac{p_{uj}}{d_l}) = \ln(\bar{\lambda}_u \cdot \frac{p_{uj}}{d_l}) + \chi_u \quad \Leftrightarrow \quad \lambda_u = \bar{\lambda}_u \cdot e^{\chi_u} \tag{12.10}$$

As seen from (12.9), this change in the multiplier adjusts all quotients from constituency u by χ_u. Since only λ_u is adjusted, quotients from all other constituencies are unchanged. Given the updated quotients, we make the assignment and calculate ρ. One iteration of the apportionment algorithm is brought to its end.

Our primary concern above was to make sure that u became rightly represented. What really matters in the big scheme of things is the development of the measure of goodness ρ. Below we analyse the impact of the adjustment on ρ. Let us first explain what happens at the seat level: We focus on the yth pair of challenging and benchmark quotient, where $y \leq \rho_u$. Let c denote the constituency the benchmark quotient comes from. Constituency u has won a seat from this constituency. The consequences for ρ of this seat transfer are: If constituency c was not overrepresented beforehand, the reduction of ρ_u by one is cancelled out by the offsetting increase in ρ_c. On the other hand, if c was overrepresented beforehand, both ρ_u and ρ_c have decreased by one and ρ thereby by two.

We move on to an analysis of the total effect of the adjustment: Constituency u has won ρ_u seats at the expense of the ρ_u smallest benchmark quotients. Constituencies which have lost seats can be classified in the following three groups:

- M^1: Constituencies which were not overrepresented beforehand.
- M^2: Constituencies which were overrepresented before the adjustment and have not become underrepresented afterwards.
- M^3: Constituencies which were overrepresented before the adjustment, but have become underrepresented afterwards.

Constituencies from M^1 have become more underrepresented after the adjustment. The total impact on ρ of seats transferred from this group to u is zero, due to the cancelling out effect described earlier. The cancelling out effect does not occur for constituencies in M^2. For each seat this group has lost to u, ρ has decreased by two. Each seat a constituency in M^3 loses until it reaches the position of being rightly represented reduces ρ by two. All seats transferred to u beyond this point result in the cancelling out effect. Hence, ρ has decreased by twice the amount of overrepresentation for M^3 beforehand.

The conclusion is that under no circumstances does ρ increase as a result of the prescribed adjustment. In fact, the only situation where ρ does not decrease is when all losing constituencies belong to M^1. The total change in ρ is found by adding the described changes caused by constituencies in M^2 and M^3. In section 12.5 we describe how the change in ρ can be calculated before the adjustment is carried out.

12.3 Downadjustment

The process of downadjusting the multiplier for an overrepresented constituency is almost similiar to the upadjustment process. In this section we review the differences between the two processes. We now assume that constituency u is overrepresented by $\rho_u > 0$ seats:

(12.11) $\rho_u = \tilde{a}_{uNH} - R_u$ where $u \in M^+$

To make $\rho_u = 0$, ρ_u quotients from constituency u must each lose a seat. The candidates in cell (u,v) are the smallest assigned quotients $q_{uva} \in Q$. We call these quotients **outgoing quotients** and denote them $q^O_{uv}(x)$. With $w > 0$ assigned quotients in cell (u,v), the xth outgoing quotient is:

(12.12) $q^O_{uv}(x) = q_{uv(w+1-x)}$ where $x \in \{1, ..., \min[(w, (\rho_u + 1)]\}$

When there are less than $(\rho_u + 1)$ assigned quotients in cell (u,v), also including the situation with no assignment, i.e. $w = 0$, we define one artificial outgoing quotient within party v: $q^{O}_{uv}(w + 1) = 50$. The background for this choice is described in the last section.

In the downadjustment context we define **benchmark quotients** as the largest unassigned quotients which do not belong to constituency u. The benchmark quotients for constituency u within party v are determined recursively as:

(12.13) $$q^{B}_{uv}(x) = \max_{s \in \{M \setminus u\}} q_{sva} \qquad \text{where } q_{sva} \in \overline{Q}$$

The distance within the xth pair of outgoing quotient and benchmark quotient is:

(12.14) $$d_{uv}(x) = q^{O}_{uv}(x) - q^{B}_{uv}(x) \qquad \text{for } x \in \{1, ..., \min[(w + 1), (\rho_u + 1)]\}$$

$d_{uv}(x)$ is non-negative and $d_{uv}(x + 1) > d_{uv}(x)$ because $q^{O}_{uv}(x + 1) > q^{O}_{uv}(x)$ and $q^{B}_{uv}(x + 1) \leq q^{B}_{uv}(x)$.

As in the upadjustment process, we determine the $(\rho_u + 1)$ smallest global distances $d_u(y)$ by (12.7) and the adjustment distance χ_u by (12.8). However, χ_u is now used to adjust quotients from constituency u downwards:

(12.15) $$q_{ujl} = \overline{q}_{ujl} - \chi_u$$

This means that the updated constituency multiplier is found as:

(12.16) $$\lambda_u = \overline{\lambda}_u \cdot e^{-\chi_u}$$

When $d_u(\rho_u + 1) > d_u(\rho_u)$, different values of the adjustment weight υ have the following consequences: With υ close to 1, the adjusted ρ_uth outgoing quotient becomes just a little bit smaller than the ρ_uth benchmark quotient, while υ in the neighbourhood of 0 results in an adjusted $(\rho_u + 1)$th outgoing quotient which barely stays above the $(\rho_u + 1)$th benchmark quotient. NB. The benchmark

quotients in the up and downadjustment cases are called challenged and incoming quotients respectively in the Pascal program in Appendix 2.

To analyse how a downadjustment changes ρ, one should move along the lines of the analysis at the end of section 12.2. Since constituency u here gives away seats, it is the constituencies which win these seats that are of interest in the analysis. The prescribed downadjustment results in a reduced or unchanged ρ.

12.4 Determination of a good value for υ

It turns out that the number of iterations to reach the optimal apportionment depends on the υ value used in formula (12.8). To identify a good value for the adjustment weight, i.e. a υ value which results in a "low" average number of iterations, we carry out a test which is conducted as follows:

We utilize the same 6 data sets, 5 bound vectors, and 4 matrix apportionment methods as in the test in section 11.5. To determine initial assignments for each combination of these three parameters, we apply the 3 initialization procedures described in section 11.3, where the target weight $\tau = 0{,}3$ is used for apportionment initialization. From each such initial assignment we solve the problem at hand with each of the 11 υ values shown in Table 12.1. The same υ value is used in every iteration. During the solution process we let representation selection decide which constituency multiplier to adjust next. Table 12.1 below summarizes the test results with constituency relaxation.

Each entry in the main body of Table 12.1 is the average number of iterations over $5 \cdot 4 \cdot 3 = 60$ cases. The lowest average for each country is shown in bold print. Based on the sum column, adjustment weights in the region 0,2 - 0,5 seem preferable, with $\upsilon = 0{,}3$ resulting in the lowest sum of averages.

Table 12.1

υ	AUT 90	ICE 91	FIN 91	DEN 84	NOR 93	SWE 91	Sum
0,99	35,75	35,75	62,37	89,83	84,05	94,18	401,93
0,9	8,57	10,80	21,47	49,25	48,55	53,03	191,67
0,8	7,30	8,42	15,12	29,47	35,00	33,65	128,96
0,7	6,47	7,62	12,90	22,87	26,08	26,53	102,47
0,6	6,67	7,48	12,27	20,23	22,48	24,58	93,71
0,5	**6,37**	**7,18**	12,30	19,43	20,85	24,00	90,13
0,4	6,92	7,35	**11,95**	19,17	20,05	**23,68**	89,12
0,3	7,03	7,33	12,12	**18,63**	**19,13**	24,28	**88,52**
0,2	7,28	7,60	12,65	18,73	19,27	25,62	91,15
0,1	8,20	8,05	13,62	21,15	20,82	27,72	99,56
0,01	13,87	11,40	16,55	22,33	28,03	36,80	128,98

Average number of iterations with constituency relaxation.

In the test we operated with an iteration limit of 100. The execution of the apportionment algorithm was abandoned when this limit was reached. All countries had cases of abandonment for $\upsilon = 0{,}99$, the three Scandinavian countries also had abandoned cases with $\upsilon = 0{,}9$, and for Norway and Denmark such cases even occurred with $\upsilon = 0{,}8$. Moreover, one Swedish case was abandoned for $\upsilon = 0{,}01$. We have counted abandoned cases as 100 iterations although the optimal apportionments were not reached yet. Some of the reported averages are therefore lower than what the real averages would have been.

We also carry out the test with the party relaxed formulation. The results of this test are presented in Table 12.2 below. We notice that every average in Table 12.2 is lower than the corresponding average in Table 12.1. Moreover, most sums are about 40% lower in Table 12.2. As in Table 12.1, the adjustment weights in the region $0{,}2 \leq \upsilon \leq 0{,}6$ result in lower sums than the other weights.

There were fewer abandoned cases with party relaxation than with constituency relaxation. In fact, all Austrian cases were solved. The other countries all had cases of abandonment for $\upsilon = 0{,}99$, but only the Scandinavian countries had abandoned cases with $\upsilon = 0{,}9$.

Table 12.2

υ	ICE 91	AUT 90	FIN 91	NOR 93	SWE 91	DEN 84	Sum
0,99	12,05	8,05	41,08	61,52	61,48	61,78	245,96
0,9	4,28	4,65	12,02	29,35	24,88	26,87	102,05
0,8	3,97	4,23	9,42	18,30	16,13	18,42	70,47
0,7	**3,88**	4,30	9,12	14,50	13,95	15,88	61,63
0,6	3,98	**4,15**	9,05	13,15	13,32	14,20	57,85
0,5	4,20	4,43	**8,70**	12,25	12,68	**14,05**	56,31
0,4	4,12	4,32	9,03	12,28	12,10	14,52	56,37
0,3	4,35	4,82	8,98	11,73	**11,92**	14,40	**56,20**
0,2	4,60	5,00	9,87	**11,65**	12,45	**14,05**	57,62
0,1	5,75	5,27	9,98	12,45	12,87	14,47	60,79
0,01	7,62	5,28	10,52	13,27	13,97	16,45	67,11

Average number of iterations with party relaxation.

We take the sum columns in both tables plus the information regarding frequency of abandoned cases into consideration when we evaluate the adjustment weights. υ = 0,99 and 0,9 score poorly on both criteria and are clearly unacceptable. The values υ = 0,8 and 0,01 should not be used either. υ = 0,1 and υ = 0,7 are better than the already mentioned weights, but inferior to weights in the region 0,2 - 0,6. In both tables υ = 0,3 results in the lowest sum of averages, though only marginally. This is the reason for our choice of υ = 0,3 as adjustment weight in the remainder of this dissertation.

12.5 Calculation of ρ-effect

At the end of section 12.2 it was explained how a multiplier adjustment changes the measure of goodness ρ. In this section we explain how the impact on ρ can be worked out prior to the adjustment. To carry out this task we need to know the under/overrepresentation of each constituency. In addition, we need information about which constituencies and parties the different benchmark quotients, and thereby the global distances, belong to.

We define the ρ-**effect** of a multiplier adjustment as the reduction in the measure of goodness ρ this adjustment will cause. ϕ_i denotes the ρ-effect of adjusting constituency i's multiplier λ_i. To shorten things, we call ϕ_i the ρ-effect for constituency i. NB. In the Pascal program in Appendix 2 the ρ-effect is called improvement and its value is only half the value here.

Algorithm 12.2 below describes an efficient way of determining the $(\rho_u + 1)$ smallest global distances plus the ρ-effect for the malrepresented constituency u. In this algorithm we use the expression challenging/outgoing quotient, which should be understood as follows: When $u \in M^-$, it refers to a challenging quotient, and when $u \in M^+$, it refers to an outgoing quotient. Moreover, we denote the opposite malrepresentation group of the one u belongs to by $\overline{M}$, so $\overline{M} = M^+$ when $u \in M^-$ and $\overline{M} = M^-$ when $u \in M^+$. During the determination process we keep track of the amount of malrepresentation for each constituency. For this purpose we define a variable called remaining malrepresentation. It is denoted $\hat{\rho}_i$ and is unconstrained in sign. We define $\hat{\rho}_i$ to be positive when constituency i is from the same malrepresentation group as u and negative when i is from the opposite malrepresentation group.

Algorithm 12.2

Step 1: This step is primarily concerned with initializations: The ρ-effect for constituency u is zero initially, i.e. we set $\phi_u = 0$. The next initialization concerns the variables for remaining malrepresentation. Based on the current assignment, they are defined as follows: $\hat{\rho}_s = -\rho_s$ for all $s \in \overline{M}$ and $\hat{\rho}_s = \rho_s$ for all $s \in \{M \setminus \overline{M}\}$. The last initialization concerns the distance numbers for distances within parties and global distances. We set: $x_j = 1$ for all parties j and $y = 1$. Then we are ready to begin the determination process. Within each party do the following: Identify the pair of first benchmark quotient and first challenging/ outgoing quotient and calculate the first distance $d_{uj}(1)$. Moreover, notice which constituency the first benchmark quotient belongs to and associate this constituency with the first distance by denoting the distance $d^i_{uj}(1)$. After all first party distances have been calculated, the first global distance $d_u(1)$ is found as the smallest of these distances.

Step 2: Let us denote the constituency and party associated with the current global distance c and e respectively. We incorporate this information in the notation of the yth (first) global distance by denoting it $d_u^{ce}(y)$ ($d_u^{ce}(1)$). This global distance has the following implications for the ρ-effect and the variables for remaining malrepresentation: If $\hat{\rho}_c < 0$, increase ϕ_u by two, otherwise keep ϕ_u unchanged. The assignment status of the benchmark quotient from constituency c and the challenging/outgoing quotient from constituency u is altered, so increase $\hat{\rho}_c$ by one and decrease $\hat{\rho}_u$ by one. If now $\hat{\rho}_c \geq 0$, benchmark quotients from c will no longer contribute to an increase in ϕ_u. The next task is the updating of distance numbers: Increase y by one (to 2) and x_e by one (to 2). Party e is without an eligible distance, so identify the x_eth (second) benchmark quotient and the x_eth (second) challenging/outgoing quotient within this party and calculate the x_eth (second) distance. Notice which constituency the benchmark quotient comes from and associate this constituency with e's new distance by denoting the distance $d_{ue}^i(x_e)$ ($d_{ue}^i(2)$). This newly calculated distance together with the current distances $d_{uj}^i(x_j)$ ($d_{uj}^i(1)$) within the other parties are candidates for the next global distance. The yth (second) global distance $d_u(y)$ ($d_u(2)$) is found as the smallest of these candidates. Step 2 is repeated until the $(\rho_u + 1)$ smallest global distances have been determined.

The comments in parentheses in step 2 refer to the first time the step is executed. Notice that ϕ_u is unaffected by the $(\rho_u + 1)$th global distance since the adjustment of λ_u will not alter the assignment status of the $(\rho_u + 1)$th benchmark quotient. It is also worth noticing that we only need to calculate $n + \rho_u$ distances to determine the $(\rho_u + 1)$ smallest global distances for constituency u. The final values of the variables for remaining malrepresentation are useful by-products of Algorithm 12.2. They tell us what the malrepresentation of each constituency will be after the adjustment of λ_u.

If one makes an arbitrary multiplier adjustment, the ρ-effect may well turn out to be negative. However, the way we conduct the adjustment process ensures that the ρ-effect is non-negative as long as the two global distances in (12.8) are unequal. The ρ-effect for a constituency cannot be larger than twice this constituency's malrepresentation. Hence, with our adjustment scheme: $0 \leq \frac{\phi_u}{2} \leq \rho_u$.

We now present ρ-**effect selection**. As the name indicates, the constituency with maximal ρ-effect is selected for the adjustment:

(12.17) $$\max_{i \in (M^- \cup M^+)} \phi_i$$

When there is a tie between constituencies from both M^- and M^+ for this position, we next look at the maximal under/overrepresentation among these constituencies. Should there still be a tie between M^- and M^+, we choose the opposite malrepresentation group of the one adjusted in the previous iteration. Finally, a tie between constituencies within the chosen malrepresentation group is broken by the rotation scheme mentioned in connection with representation selection.

The advantage of ρ-effect selection is that it enables us to reduce ρ as much as possible in each iteration. This is especially important in situations where only one or a few of the possible adjustments will reduce ρ. Nothing comes for free, the drawback of ρ-effect selection is that it may require a lot more work than representation selection. With representation selection there is no need for determining ρ-effect. Furthermore, the determination of global distances is restricted to one constituency. Because there does not exist a direct relationship between amount of malrepresentation and size of ρ-effect, one may have to determine global distances and ρ-effect for several constituencies to implement ρ-effect selection. A way of scheduling these calculations is to start with constituencies with maximal malrepresentation and continue, if necessary, with constituencies for which ρ_i is greater than the current maximal ρ-effect. We compare the performance of ρ-effect and representation selection in the test in chapter 14.

12.6 Practical problems

In this section we describe some practical problems we encountered while programming the apportionment algorithm and the choices they led to. The three main topics treated are:

1. How many multipliers to adjust in each iteration.
2. Why both up and downadjustments are made.
3. The reasons for defining artificial quotients.

We have chosen to adjust only one constituency multiplier in each iteration. Example 12.1 is an illustration of what may happen if one makes a simultaneous adjustment of two or more multipliers:

Example 12.1

A and B are two constituencies which both are underrepresented by one seat, i.e. $\rho_A = 1$ and $\rho_B = 1$. Moreover, their smallest global distances, $d_A^{ce}(1)$ and $d_B^{se}(1)$ respectively, occur within the same party. To study the impact of a simultaneous adjustment of λ_A and λ_B, we break the adjustment in two by first adjusting λ_A. This adjustment leads to A winning a seat at the expense of its benchmark quotient, i.e. constituency $c \in \{M \setminus A\}$ loses a seat. Thereafter λ_B is adjusted according to the benchmark quotient for constituency B. This quotient comes from constituency $s \in \{M \setminus B\}$. There are two possibilities regarding the status of this benchmark quotient: If $s \in \{M \setminus (A \cup B)\}$, A and B had the same benchmark quotient initially, i.e. $s = c$. In this case the quotient has become unassigned after the adjustment of λ_A. The other possibility is that the benchmark quotient for B came from constituency A, i.e. $s = \{A\}$. Then the adjustment of λ_A has increased the quotient's value. In either case, the adjustment of λ_B bases itself on a false benchmark quotient. This may result in a too small adjustment distance χ_B for B to win a seat. Hence, simultaneous adjustment of λ_A and λ_B does not guarantee that both constituencies become rightly represented afterwards. Both constituencies have won a seat if the position of A's adjusted challenging quotient within

party e was better than number C_e before the adjustment of λ_B and B's challenging quotient was adjusted by enough to be placed as number C_e or better.

Simultaneous multiplier adjustments based on the principles in sections 12.2 or 12.3 do not increase ρ, so why have we elected not to employ such adjustments? The explanation is as follows: As the number of iterations increases without the optimal apportionment being found, quotients cluster around the value of the C_jth quotient within each party. Simultaneous multiplier adjustments speed up this process because of many futile adjustments. The clustering of quotients makes the global distances $d_u(\rho_u)$ and $d_u(\rho_u + 1)$ and thereby the adjustment distance χ_u very small. Due to the computer's somewhat inaccurate treatment of numbers, χ_u may eventually be perceived as zero, with the result that the program converges towards an infeasible apportionment. We have reserved more storage space for real numbers in the computer to better the accuracy. Nevertheless, our program of the apportionment algorithm is still better off without simultaneous multiplier adjustments.

False benchmarks occur when one or more global distances for two or more malrepresented constituencies come from the same party. If the smallest global distances $d_i(1)$ for a group of malrepresented constituencies come from different parties and we only adjust these constituencies for one seat each, it is possible to adjust their multipliers simultaneously without relying on false benchmarks. In practice, the large parties tend to have a large share of the smallest global distances, thereby restricting the number of constituencies which qualify for such a group. The fact that we adjust constituencies for more than one seat when $\rho_i > 1$ makes it even harder to keep track of real benchmark quotients.

We never adjust for both under and overrepresentation in the same iteration. Such simultaneous multiplier adjustments run into even more severe complications than those described above. However, when the apportionment algorithm was tested on data from the Swedish election in 1991, it became evident that it is preferable to adjust both under and overrepresented constituencies compared to only adjusting underrepresented constituencies:

Example 12.2

The interesting situation occurred when $CP_{0,01}$ was used for the matrix apportionment. To illustrate what the $CP_{0,01}$ matrix apportionment looks like, we give a description of the SD apportionment. SD (CP_0), which is a little bit more extreme than $CP_{0,01}$, let as few cells as possible be left with zero representatives. Now to the other election parameters of interest: There were 28 constituencies. The smallest of these, Gotland, was awarded 2 seats. Of the parties participating in the election, four got more than 28 representatives. With $CP_{0,01}$ and no initialization, the initial assignment spread the first 28 representatives within each of these parties evenly between the 28 constituencies. Thus, Gotland was overrepresented by 2 seats initially. To make Gotland rightly represented, and thereby take an important step towards the optimal apportionment, it was much more efficient to decrease the multiplier for Gotland once than to increase the multipliers for other constituencies several times.

Example 12.2 is the reason why we started making both up and downadjustments of multipliers. Section 12.5 has supplied us with another argument: A constituency with maximal ρ-effect belongs to either M^- or M^+. One must therefore be free to choose which kind of adjustment to make to decrease ρ as much as possible in every iteration.

The next topic we deal with is how to schedule up and downadjustments. One possibility is to alternate on a regular basis, e.g. by adjusting for underrepresentation in odd numbered iterations and for overrepresentation in even numbered iterations. As explained in sections 12.1 and 12.5, our priority list regarding which constituency to adjust is based on maximal ρ-effect and/or maximal malrepresentation. Only when constituencies from both M^- and M^+ are tied on the last criterion do we alternate between the constituency groups on a regular basis.

Regarding choice of value for the adjustment weight υ, it is possible to define separate values for up and downadjustments. Furthermore, one could operate with different values during the solution process. We have opted for a common υ value to keep it simple.

As described in sections 12.2 and 12.3, we have introduced artificial benchmark quotients for the upadjustment case and artificial outgoing quotients for the downadjustment case. In all situations where an artificial quotient comes into effect, it does so via the $(\rho_u + 1)$th global distance. Thus, one artificial global distance for constituency u would have been enough. We have defined up to one artifical quotient within each party, because this is a suitable way of making sure that an artificial $(\rho_u + 1)$th global distance is available when needed.

Example 12.3 illustrates why artificial benchmark quotients are needed in situations where a constituency shall have a positive number of seats, but currently are assigned zero seats:

Example 12.3

We face an equality constrained problem with two parties, labelled A and B, and two constituencies, labelled 1 and 2. The election data are as follows: Both constituency bounds are positive, i.e. $R_1 > 0$ and $R_2 > 0$. Party A, which shall have $C_A > 0$ seats, receives votes only in constituency 1, i.e. $p_{1A} > 0$ and $p_{2A} = 0$. The apportionment to party B is restricted to $C_B = 1$ seat. Party B receives votes in both constituencies, but more in the first than in the second constituency, i.e. $p_{1B} > p_{2B} > 0$. Finally, the initial constituency multipliers are equal $\lambda_1 = \lambda_2$. From this information it is clear that constituency 2 is not assigned any seat initially. We utilize the two smallest global distances for constituency 2, i.e. $d_2(1)$ and $d_2(2)$, in the calculation of the adjustment distance χ_2. $d_2(1)$ is easily found within party B. However, to be able to calculate $d_2(2)$, an artificial benchmark quotient for constituency 2 within party B must be defined. The optimal apportionment, which is reached after one iteration, is: $a_{1A} = C_A$, $a_{2B} = 1$, and $a_{1B} = a_{2A} = 0$.

Here follows the explanation of why artificial outgoing quotients were introduced: Contrary to earlier assumptions, we have allowed upper bounds equal to zero in the Pascal program. The reason for this decision is as follows: In the test in chapter 14 we operate with three bound vectors; the real bound vector plus two bound vectors which include all parties with a vote percentage of at least 1%. Because election laws normally exclude parties with about 1% support from

getting represented, the real bound vector often includes some parties with bounds equal to zero. To solve the party relaxed version of the matrix apportionment problems, we transpose the election data such that our existing program based on constituency relaxation can be utilized. Then party bounds are converted to constituency bounds. Now, consider the following situation: Constituency u is currently assigned a positive number of seats. However, its constituency bound is equal to zero, i.e. $R_u = 0$. Then an artificial outgoing quotient is needed to calculate the $(\rho_u + 1)$th global distance.

It is important that artificial quotients are high, but they should not be too high. The explanation is based on the assumption that an artificial quotient comes into effect, i.e. is needed for the calculation of the $(\rho_u + 1)$th global distance. A too high quotient value will result in the adjustment distance χ_u becoming very large. From (12.10) and (12.16) we see that this leads to an enormous constituency multiplier in the upadjustment case and an extremely small constituency multiplier in the downadjustment case. The problem with an extremely small multiplier is that the computer fails to distinguish it from 0. Since the natural logarithm of 0 is undefined, the program crashes. On the other hand, the value of an artificial quotient cannot be set too low either. The value must be so high that it results in artificial global distances which are larger than the ρ_uth global distance in all imaginable situations.

As stated in sections 12.2 and 12.3, we have chosen the value 50 for the artificial quotients. This value has stood its test in numerous computer runs. From the formula for a quotient, $\ln(\lambda_i \cdot \frac{p_{ij}}{d_j})$, we see that the chosen value of 50 corresponds to a cell vote of $p_{ij} = \frac{d_j}{\lambda_i} \cdot e^{50}$.

Chapter 13: Algorithm example

This chapter contains a numerical example which illustrates how the iterative apportionment algorithm described in chapter 12 works. The chosen matrix apportionment problem is presented in the first section. Section 13.2 shows how the assignment process described in section 11.1 is carried out at the initial stage. The next two sections, section 13.3 and section 13.4, show the necessary calculations of distances and ρ-effect for a downadjustment and an upadjustment respectively. In the final section we reach the optimal apportionment.

13.1 The problem

We utilize vote data from the Icelandic general election in 1995. The whole election situation consists of 8 constituencies × 6 parties with 63 seats for distribution. To keep it simple, the 3 × 3 subproblem in Table 13.1 has been picked out.

Table 13.1

ij	B	D	G	Votes	Seats
Rev	9743	27736	9440	46919	12
NoE	6015	4606	2741	13362	5
Aul	3668	1760	1257	6685	5
Votes	19426	34102	13438	66966	
Seats	7	11	4		22

Vote matrix, house size, and constituency (row) and party (column) constraints for a 3 × 3 subproblem of the Icelandic general election in 1995.

This subproblem consists of the three largest parties in the 1995 election, Progressive Party (B), Independence Party (D), and People's Alliance (G) and their votes in Reykjavík (Rev), Nordurland eystra (NoE), and Austurland (Aul). Reykjavík is the largest, Nordurland eystra is the third largest, while Austurland is one of the small constituencies in Iceland. The house size plus the row and column constraints given in Table 13.1 show the apportionment to the 3 × 3 subproblem when HA is used as matrix apportionment method for the whole 8 × 6 election situation. We are going to solve the matrix apportionment problem in Table 13.1 with HA (CP_1) as matrix apportionment method. The chosen starting position for the apportionment algorithm is no initialization, i.e. $\mathring{\lambda}_{Rev} = \mathring{\lambda}_{NoE} = \mathring{\lambda}_{Aul} = 1$. A piece of information, it takes 3 iterations to solve the problem from this starting position, while both quota ratio and apportionment initialization would have solved it directly. Now to the solution of the problem:

13.2 Initial stage

At the initial stage we check whether the initial multipliers have produced the optimal apportionment, i.e. whether $\mathring{\rho} = 0$. The first task is the determination of the initial assignment: With the initial constituency multipliers and the divisor series for HA, we calculate the quotients $\ln(\lambda_i \cdot \frac{p_{ij}}{d_l})$ we find necessary. An example of such a calculation: The third quotient for party B in constituency Rev, which we denote Rev.B.3 in tables and $q_{Rev.B.3}$ elsewhere, is $\ln(1 \cdot \frac{9743}{3}) \approx 8{,}086$. Within each party we sort the calculated quotients in descending order. Table 13.2, Table 13.3, and Table 13.4 below present the sorted party lists for B, D, and G respectively.

Table 13.2

Place	Quotient	Value	Classification	Distance	Global no.
1	Rev.B. 1	9,184			
2	NoE.B. 1	8,702			
3	Rev.B. 2	8,491	3rd Outgoing	1,175	
4	Aul.B. 1	8,207			
5	Rev.B. 3	8,086	2nd Outgoing	0,572	4th
6	NoE.B. 2	8,009			
7	Rev.B. 4	7,798	1st Outgoing	0,195	2nd
8	NoE.B. 3	7,603	1st Benchmark		
9	Rev.B. 5	7,575			
10	Aul.B. 2	7,514	2nd Benchmark		
11	Rev.B. 6	7,393			
12	NoE.B. 4	7,316	3rd Benchmark		
13	Rev.B. 7	7,238			
14	Aul.B. 3	7,109			

Sorting of quotients within party B + classification of quotients, distances between the xth outgoing and xth benchmark quotient, and global ranking of these distances when a down-adjustment of λ_{Rev} is considered.

As the tables reveal, the sorted party lists are dominated by quotients from Rev. The use of HA, which favours large entities, contributes to this domination.

Table 13.3

Place	Quotient	Value	Classification	Distance	Global no.
7	NoE.D. 1	8,435			
8	Rev.D. 7	8,285	4th Outgoing	1,236	
9	Rev.D. 8	8,151	3rd Outgoing	0,814	5th
10	Rev.D. 9	8,033	2nd Outgoing	0,560	3rd
11	Rev.D. 10	7,928	1st Outgoing	0,186	1st
12	Rev.D. 11	7,833			
---	-------	----			
14	NoE.D. 2	7,742	1st Benchmark		
---	-------	----			
18	Aul.D. 1	7,473	2nd Benchmark		
---	-------	----			
22	NoE.D. 3	7,337	3rd Benchmark		
---	-------	----			
29	NoE.D. 4	7,049	4th Benchmark		
---	-------	----			
37	Aul.D. 2	6,780			

Sorting of quotients within party D + classification of quotients, distances between the xth outgoing and xth benchmark quotient, and global ranking of these distances when a down-adjustment of λ_{Rev} is considered.

The dotted lines in the left part of Table 13.3 and Table 13.4 symbolize quotients which are without interest in the demonstration of the apportionment algorithm.

Table 13.4

Place	Quotient	Value	Classification	Distance	Global no.
1	Rev.G. 1	9,153			
2	Rev.G. 2	8,460			
3	Rev.G. 3	8,054	1st Outgoing	0,831	6th
4	NoE.G. 1	7,916			
5	Rev.G. 4	7,766			
---	-------	----			
8	NoE.G. 2	7,223	1st Benchmark		
9	Rev.G. 7	7,207			
10	Aul.G. 1	7,136			

Sorting of quotients within party G + classification of quotients, distances between the xth outgoing and xth benchmark quotient, and global ranking of these distances when a down-adjustment of λ_{Rev} is considered.

Only the three columns in the left part of Table 13.2 - Table 13.4 are of interest at the present stage. In each table the dividing line between assigned and unassigned quotients, C_j, is marked in bold. The constituencies are assigned seats according to their number of quotients above these dividing lines. This results in the assignment presented in the left part of Table 13.5:

Table 13.5

	Parties					Representation		
	B	D	G	Total	Constraint	Under	Over	ρ
Rev	4	10	3	17	12	–	5	
NoE	2	1	1	4	5	1	–	
Aul	1	0	0	1	5	4	–	
	7	11	4			5	5	10

Number of seats assigned to the constituencies within the three parties, malrepresentation, and measure of goodness with the current constituency multipliers: $\lambda_{Rev} = \lambda_{NoE} = \lambda_{Aul} = 1$.

The right part of Table 13.5 tells us that Rev is overrepresented by 5 seats while NoE and Aul are underrepresented by 1 and 4 seats respectively. This adds up to an initial measure of goodness of $\mathring{\rho} = 10$. Since $\rho > 0$, the current assignment is not the optimal apportionment. The apportionment algorithm, i.e. Algorithm 12.1, must therefore be employed. In the demonstration of this algorithm we apply ρ-effect selection, but the consequences of using representation selection

instead are also described.

13.3 First iteration

From Table 13.5 we see that Rev is most malrepresented. Hence, representation selection, see step 2 in Algorithm 12.1, would have chosen to downadjust the multiplier for Rev. It turns out that λ_{Rev} also is the constituency multiplier to be adjusted by ρ-effect selection. To determine global distances and ρ-effect for Rev, we employ Algorithm 12.2. Below we describe the determination process thoroughly. The distances calculated during this process are also presented in the right part of Table 13.2 - Table 13.4 above.

We begin with initializations: The ρ-effect for Rev is zero initially $\phi_{Rev} = 0$, while all distance numbers are equal to one, i.e. $x_j = 1$ for all parties j and $y = 1$. All constituencies are malrepresented at the present stage, with $M^+ = \{Rev\}$ and $M^- = \{NoE, Aul\}$. Since Rev is overrepresented, the group of constituencies with opposite malrepresentation is $\overline{M} = M^-$. We use the malrepresentation data in Table 13.5 to initialize the variables for remaining malrepresentation: $\hat{\rho}_{NoE} = -1$, $\hat{\rho}_{Aul} = -4$, and $\hat{\rho}_{Rev} = 5$.

Next on the agenda is identification of outgoing and benchmark quotients for Rev. In this process we utilize the sorted party lists in Table 13.2 - Table 13.4. Within each party we find the smallest quotient from Rev which is assigned a seat. This is the first outgoing quotient. The first outgoing quotient within party B is the fourth quotient in the cell for constituency Rev and party B. It is denoted $q^{O}_{Rev.B}(1) = q_{Rev.B.4}$ Within parties D and G the first outgoing quotients are $q_{Rev.D.10}$ and $q_{Rev.G.3}$ respectively. It is time to identify the first benchmark quotients. Within each party the first benchmark quotient is the largest unassigned quotient from another constituency than Rev. The first benchmark quotient within party B is the third quotient from NoE. It is denoted $q^{B}_{Rev.B}(1) = q_{NoE.B.3}$. The first benchmark quotients within D and G are $q_{NoE.D.2}$ and $q_{NoE.G.2}$

respectively. In Table 13.2 - Table 13.4 outgoing and benchmark quotients are labelled in the column to the right of the quotient values. We go on to calculate the first distances within the three parties. They are as follows: $d^{NoE}_{Rev.B}(1) = q^{O}_{Rev.B}(1) - q^{B}_{Rev.B}(1) = q_{Rev.B.4} - q_{NoE.B.3} \approx 7{,}798 - 7{,}603 = 0{,}195$ within B, $d^{NoE}_{Rev.D}(1) = q_{Rev.D.10} - q_{NoE.D.2} \approx 0{,}186$ within D, and finally $d^{NoE}_{Rev.G}(1) = q_{Rev.G.3} - q_{NoE.G.2} \approx 0{,}831$ within G. These and other distances are shown to the right of the outgoing quotients in Table 13.2 -Table 13.4 above. We observe that the smallest first distance is within D. This is the first global distance in connection with the downadjustment of λ_{Rev}. $d^{NoE.D}_{Rev}(1) \approx 0{,}186$. The notation also shows within which party, D, and for which other constituency, NoE, the distance occurs. In Table 13.3 the distance is marked "1st" in the "Global no." column.

We calculate global distances to determine how large a multiplier adjustment must be to lead to an exchange of unassigned and assigned quotients. The beneficiary within party D in connection with the first global distance is the second quotient from NoE. Since Rev is overrepresented and NoE currently is underrepresented, a downadjustment of λ_{Rev} such that $q_{Rev.D.10}$ becomes smaller than $q_{NoE.D.2}$ will reduce ρ. Thus, ϕ_{Rev} is increased by two to 2. At the same time the variable for remaining malrepresentation for NoE is increased by one and becomes zero, i.e. $\hat{\rho}_{NoE} = 0$, while the variable for Rev is decreased by one to $\hat{\rho}_{Rev} = 4$. These values tell us that a downadjustment based on the first global distance would result in NoE becoming rightly represented and Rev becoming overrepresented by 4 seats. The first global distance belongs to party D, so both y and x_D are increased by one to 2. Before we can determine the second global distance, the second distance within party D must be calculated. From Table 13.3 we see that the first outgoing quotient within D is placed as number 11. The second outgoing quotient within D is the smallest higher placed quotient from Rev, so $q^{O}_{Rev.D}(2) = q_{Rev.D.9}$. Furthermore, the first benchmark quotient within D is in 14th place. The largest lower placed quotient from another constituency than Rev is $q_{Aul.D.1}$. This is the second benchmark quotient within D: $q^{B}_{Rev.D}(2) = q_{Aul.D.1}$. From these second quotients, we calculate the second distance within party D: $d^{Aul}_{Rev.D}(2) = q_{Rev.D.9} - q_{Aul.D.1} \approx 8{,}033 - 7{,}473 = 0{,}560$. This distance together

with the first distances within B and G, $d_{Rev.B}(1) \approx 0{,}195$ and $d_{Rev.G}(1) \approx 0{,}831$, are candidates for the second global distance. The smallest of these is the one within party B, so the second global distance is $d_{Rev}^{NoE.B}(2) \approx 0{,}195$.

We move on to the determination of the third global distance. The actual distance numbers are increased by one each, y to 3 and x_B to 2. Since the second global distance involves a benchmark quotient from NoE and $\hat{\rho}_{NoE} = 0$, ρ will not be reduced by this benchmark quotient becoming assigned. Thus, $\phi_{Rev} = 2$ still. In our remaining malrepresentation account we increase $\hat{\rho}_{NoE}$ by one to 1 and decrease $\hat{\rho}_{Rev}$ by one to 3. $\hat{\rho}_{NoE} = 1$ tells us that a downadjustment of λ_{Rev} based on the second global distance would result in NoE becoming overrepresented by 1 seat. The next task is to find the second pair of quotients within party B and calculate the distance between them. The process of finding such quotients should now be familiar, so we present the second distance within B straight away: $d_{Rev.B}^{Aul}(2) = q_{Rev.B.3} - q_{Aul.B.2} \approx 0{,}572$. This distance is larger than the second distance within party D, which is the smallest of the other distances. Thus, the third global distance is $d_{Rev}^{Aul.D}(3) \approx 0{,}560$. The process of determining the fourth and fifth global distance is described briefly below. From the description above it should be clear how the different parts of the process are carried out.

The benchmark quotient involved in the third global distance comes from Aul. Since $\hat{\rho}_{Aul} = -4 < 0$, the ρ-effect is increased by 2 to $\phi_{Rev} = 4$. The variables for remaining malrepresentation change as follows: $\hat{\rho}_{Aul}$ is increased to -3, while $\hat{\rho}_{Rev}$ is decreased to 2. y and x_D are increased to 4 and 3 respectively. The third distance within party D is $d_{Rev.D}^{NoE}(3) = q_{Rev.D.8} - q_{NoE.D.3} \approx 0{,}814$. By comparing the three available distances, we see that the second distance within party B is the fourth global distance: $d_{Rev}^{Aul.B}(4) \approx 0{,}572$.

$\hat{\rho}_{Aul} = -3 < 0$, so ϕ_{Rev} is increased to 6. Furthermore, $\hat{\rho}_{Aul}$ is increased to -2 and $\hat{\rho}_{Rev}$ is decreased to 1. We increase y to 5, and x_B to 3, so it is time to calculate the third distance within party B. It is $d_{Rev.B}^{NoE}(3) = q_{Rev.B.2} - q_{NoE.B.4} \approx 1{,}175$. The fifth global distance is the third distance within party D, calculated in the preceding paragraph. Thus, $d_{Rev}^{NoE.D}(5) \approx 0{,}814$.

Since $\hat{\rho}_{NoE} = 1 \geq 0$, the ρ-effect remains at $\phi_{Rev} = 6$. $\hat{\rho}_{NoE}$ is increased to 2, while $\hat{\rho}_{Rev}$ is decreased to 0. Because Rev was overrepresented by 5 seats at the start and we now have determined five global distances, the final ρ-effect for Rev is $\phi_{Rev} = 6$. The downadjustment of λ_{Rev} calculated below will therefore reduce the measure of goodness to $\rho = 10 - 6 = 4$. Furthermore, the final values of the variables for remaining malrepresentation, $\hat{\rho}_{Rev} = 0$, $\hat{\rho}_{NoE} = 2$, and $\hat{\rho}_{Aul} = -2$, tell us that Rev will be rightly represented, NoE will be overrepresented by 2 seats, and Aul will be underrepresented by 2 seats after this adjustment. From Table 13.8 below, which shows the assignment after the first iteration, we have the chance to verify that these postulates are correct. To calculate the adjustment distance for Rev, we also need to know the sixth global distance. y is therefore increased to 6 and x_D to 4. The fourth distance within party D is $d^{NoE}_{Rev.D}(4) = q_{Rev.D.7} - q_{NoE.D.4} \approx 1{,}236$. This distance together with $d^{NoE}_{Rev.B}(3) \approx 1{,}175$ and $d^{NoE}_{Rev.G}(1) \approx 0{,}831$ are candidates for the sixth global distance.The first distance within party G is smallest, so $d^{NoE.G}_{Rev}(6) \approx 0{,}831$.

Then we are ready to calculate the adjustment distance for Rev from equation (12.8). Since $\upsilon = 0{,}3$ has been chosen as the value of the adjustment weight, the adjustment distance becomes $\chi_{Rev} = 0{,}3 \cdot 0{,}814 + (1 - 0{,}3) \cdot 0{,}831 \approx 0{,}8259$. Equation (12.16) then gives a new multiplier value of $\lambda_{Rev} = 1 \cdot \mathbf{e}^{-0{,}8259} \approx 0{,}4378$. NB. It is sometimes necessary to operate with more decimal places than we do in this example.

χ_{Rev} is only of interest if we decide to adjust λ_{Rev}. Before this decision is made, the ρ-effects for NoE and Aul should also be calculated, i.e. Algorithm 12.2 should also be employed for these constituencies. As told at the start of the section, ρ-effect selection picks λ_{Rev} as the multiplier to be adjusted. Here follows a brief explanation of why Rev is picked: Since NoE is underrepresented by one in the assignment in Table 13.5, the maximal attainable ρ-effect for NoE is 2, which is less than the already calculated $\phi_{Rev} = 6$. Hence, it is not worthwhile calculating the ρ-effect for NoE, although our Pascal program still does. Calculation of ρ-effect is only necessary for Aul, where the maximal attainable ρ-effect is $2 \cdot \rho_{Aul} = 8$. As shown in Table 13.6 below, $\phi_{Aul} = 4$, which is lower than ϕ_{Rev}.

The data needed to determine global distances and ρ-effects for NoE and Aul are available in Table 13.2 - Table 13.4. For a thorough demonstration of calculations regarding upadjustments, we refer to the next section. Global distance data for all three malrepresented constituencies are presented in Table 13.6 below. Each row in these sorted lists shows the global distance and the benchmark quotient involved in it. In the rightmost column it is shown how the inclusion (exclusion in the upadjustment context) of this benchmark quotient in (from) the assignment will affect the measure of goodness ρ.

Table 13.6

Global distances for downadjustment of Rev

No.	Benchmark quotient	Distance	ρ – effect
1	NoE.D. 2	0,186	2
2	NoE.B. 3	0,195	0
3	Aul.D. 1	0,560	2
4	Aul.B. 2	0,572	2
5	NoE.D. 3	0,814	0
6	NoE.G. 2	0,831	–
			6

Global distances for upadjustment of NoE

No.	Benchmark quotient	Distance	ρ – effect
1	Rev.D. 10	0,186	2
2	Rev.B. 4	0,195	–
			2

Global distances for upadjustment of Aul

No.	Benchmark quotient	Distance	ρ – effect
1	Rev.B. 4	0,284	2
2	Rev.D. 10	0,455	2
3	NoE.G 1	0,780	0
4	NoE.B 2	0,900	0
5	Rev.D. 9	1,253	–
			4

Global distances, benchmark quotients, and ρ-effect for each of the three malrepresented constituencies.

With the new multiplier set $\lambda_{Rev} = 0{,}4378$, $\lambda_{NoE} = \lambda_{Aul} = 1$, we recalculate the quotients from Rev. The new quotients are equal to the old ones minus 0,8259. Thereafter we sort the three party lists in descending order. The quotients of interest in the further calculations are included in Table 13.7 below:

Table 13.7

Party B

Place	Quotient	Value	Classification	Distance	Global no.
4	NoE.B. 2	8,009			
5	Rev.B. 2	7,665	2nd Benchmark	0,844	
6	NoE.B. 3	7,603	1st Benchmark	0,494	2nd
7	Aul.B. 2	7,514			
8	NoE.B. 4	7,316			
9	Rev.B. 3	7,260			
10	Aul.B. 3	7,109	1st Challenging		
---	------	----			
14	Aul.B. 4	6,821	2nd Challenging		

Party D

Place	Quotient	Value	Classification	Distance	Global no.
7	NoE.D. 2	7,742			
8	Rev.D. 6	7,613			
9	Aul.D. 1	7,473			
10	Rev.D. 7	7,459			
11	NoE.D. 3	7,337	1st Benchmark	0,557	3rd
12	Rev.D. 8	7,325			
13	Rev.D. 9	7,207			
---	------	----			
20	Aul.D. 2	6,780	1st Challenging		

Party G

Place	Quotient	Value	Classification	Distance	Global no.
2	NoE.G. 1	7,916			
3	Rev.G. 2	7,634	2nd Benchmark	1,191	
4	Rev.G. 3	7,228	1st Benchmark	0,092	1st
5	NoE.G. 2	7,223			
6	Aul.G. 1	7,136	1st Challenging		
---	------	----			
12	Aul.G. 2	6,443	2nd Challenging		

Sorting of quotients within the three parties after the first iteration + classification of quotients, distances between the xth benchmark and xth challenging quotient, and global ranking of these distances when an upadjustment of λ_{Aul} is considered.

Based on the party lists above, we find the assignment after the first iteration:

Table 13.8

	Parties					Representation		
	B	D	G	Total	Constraint	Under	Over	ρ
Rev	2	7	3	12	12	0	0	
NoE	3	3	1	7	5	–	2	
Aul	2	1	0	3	5	2	–	
	7	11	4			2	2	4

Number of seats assigned to the constituencies within the three parties, malrepresentation, and measure of goodness with the multiplier set: $\lambda_{Rev} = 0{,}4378$, $\lambda_{NoE} = \lambda_{Aul} = 1$.

13.4 Second iteration

Since $\rho = 4$ at the moment, another iteration is needed. It turns out that both representation selection and ρ-effect selection choose to upadjust the multiplier for Aul. We therefore calculate global distances and ρ-effect for Aul in this section. A difference compared with the downadjustment of λ_{Rev} in the first iteration is that we now deal with challenging quotients instead of outgoing quotients. Furthermore, a benchmark quotient in the upadjustment context is a quotient which currently is assigned a seat.

We begin with the initializations: $\phi_{Aul} = 0$, $y = 1$, and $x_j = 1$ for all j at the start. $M^- = \{\text{Aul}\}$, $M^0 = \{\text{Rev}\}$, and $\overline{M} = M^+ = \{\text{NoE}\}$, so the initial values of the variables for remaining malrepresentation are $\hat{\rho}_{Aul} = 2$, $\hat{\rho}_{Rev} = 0$, and $\hat{\rho}_{NoE} = -2$. Our focus in the following is on challenging quotients from Aul trying to surpass benchmark quotients from Rev and NoE. The first challenging quotient within each of the three parties is the largest unassigned quotient from Aul. From Table 13.7 we see that the first challenging quotient within party B is the third quotient from Aul: $q^{C}_{Aul.B}(1) = q_{Aul.B.3}$. Moreover, the first challenging quotients within D and G are $q_{Aul.D.2}$ and $q_{Aul.G.1}$ respectively. Within each party, the first benchmark quotient is the smallest assigned quotient from another constituency than Aul. The sorted party list for B in Table 13.7 tells us that the second quotient from Aul is the smallest assigned quotient within party B. However, this

quotient is from the upadjustment constituency and is not a benchmark quotient. The quotient in 6th place is from NoE and is therefore the first benchmark quotient within B: $q^{B}_{Aul.B}(1) = q_{NoE.B.3}$. Similarly, the first benchmark quotients within D and G are $q_{NoE.D.3}$ and $q_{Rev.G.3}$ respectively. In the upadjustment context we calculate distances between benchmark quotients and challenging quotients. The first distances are: $d^{NoE}_{Aul.B}(1) = q^{B}_{Aul.B}(1) - q^{C}_{Aul.B}(1) = q_{NoE.B.3} - q_{Aul.B.3} \approx 7{,}603 - 7{,}109 \approx 0{,}494$ within B, $d^{NoE}_{Aul.D}(1) = q_{NoE.D.3} - q_{Aul.D.2} \approx 0{,}557$ within D, and $d^{Rev}_{Aul.G}(1) = q_{Rev.G.3} - q_{Aul.G.1} \approx 0{,}092$ within G. These distances are shown to the right of the first benchmark quotients in Table 13.7. The smallest distance is the one within party G, so the first global distance in connection with the upadjustment of λ_{Aul} is $d^{Rev.G}_{Aul}(1) \approx 0{,}092$.

An adjustment of λ_{Aul} such that $q_{Aul.G.1}$ just surpasses $q_{Rev.G.3}$ would not reduce ρ because Rev is rightly represented at the moment. Thus, ϕ_{Aul} remains at zero, $\hat{\rho}_{Rev}$ is increased by one to 1, while $\hat{\rho}_{Aul}$ is decreased by one to 1. We then increase the distance numbers x_G and y by one to 2 and determine the second distance within G: Since Aul is not assigned any seat within party G, the second challenging quotient is $q^{C}_{Aul.G}(2) = q_{Aul.G.2}$ Moreover, the second benchmark quotient within G is the smallest assigned quotient from Rev or NoE with a placing higher than 4th. Hence, $q^{B}_{Aul.G}(2) = q_{Rev.G.2}$ and the second distance within G is $d^{Rev}_{Aul.G}(2) = q_{Rev.G.2} - q_{Aul.G.2} \approx 1{,}191$. The first distance within party B is the smallest of the three candidates for the second global distance, so $d^{NoE.B}_{Aul}(2) \approx 0{,}494$.

An upadjustment of λ_{Aul} such that $q_{Aul.B.3}$ just surpasses $q_{NoE.B.3}$ will reduce ρ since NoE is overrepresented by two seats beforehand. The ρ-effect for Aul is therefore increased by two to $\phi_{Aul} = 2$. Moreover, the variables for remaining malrepresentation are changed as follows: $\hat{\rho}_{NoE}$ is increased to -1, while $\hat{\rho}_{Aul}$ is decreased to 0. Since $\hat{\rho}_{Aul} = 0$ now, $\phi_{Aul} = 2$ is the final ρ-effect for Aul. The upadjustment of λ_{Aul} calculated below will therefore reduce the measure of goodness to $\rho = 4 - 2 = 2$. $\hat{\rho}_{Aul} = 0$, $\hat{\rho}_{NoE} = -1$, and $\hat{\rho}_{Rev} = 1$ tell us that Aul will be rightly represented, NoE will be overrepresented by 1 seat, and Rev will be underrepresented by 1 seat after this adjustment. The reason why a positive

(negative) $\hat{\rho}$ means underrepresentation (overrepresentation) in this case is that Aul is underrepresented at the start of the determination process. The third global distance remains to be determined so we increase y to 3 and x_B to 2. We then find the second distance within B: $d^{Rev}_{Aul.B}(2) = q_{Rev.B.2} - q_{Aul.B.4} \approx 0{,}844$. Of the three candidates for the third global distance, the first distance within party D is the smallest, so $d^{NoE.D}_{Aul}(3) \approx 0{,}557$. Equation (12.8) then gives the following adjustment distance: $\chi_{Aul} = 0{,}3 \cdot 0{,}494 + (1 - 0{,}3) \cdot 0{,}557 \approx 0{,}5381$. Since we deal with an upadjustment, we employ equation (12.10), which gives a new multiplier value of $\lambda_{Aul} = 1 \cdot e^{0{,}5381} \approx 1{,}7127$. The lower part of Table 13.9 below summarizes the most interesting data from the determination process above, while the upper part of the table shows the same kind of data for a possible downadjustment of λ_{NoE}.

Table 13.9

Global distances for downadjustment of NoE

No.	Benchmark quotient	Distance	ρ – effect
1	Rev.D. 8	0,012	0
2	Rev.B. 3	0,343	0
3	Rev.D. 9	0,535	–
			0

Global distances for upadjustment of Aul

No.	Benchmark quotient	Distance	ρ – effect
1	Rev.G. 3	0,092	0
2	NoE.B. 3	0,494	2
3	NoE.D. 3	0,557	–
			2

Global distances, benchmark quotients, and ρ-effect for each of the two malrepresented constituencies.

From Table 13.9 we see that $\phi_{NoE} = 0$, which makes it clear why ρ-effect selection chooses to adjust λ_{Aul}. In fact, Aul would also have been selected by representation selection; $\rho_{NoE} = \rho_{Aul}$ after the first iteration, but since there was a downadjustment in the previous iteration, our tie breaking rule says that there shall be an upadjustment in this iteration. It is not unusual for a ρ-effect to be equal to zero, like ϕ_{NoE} here. Sometimes none of the possible adjustments will reduce the measure of goodness.

After the adjustment of λ_{Aul}, the multiplier set is $\lambda_{Rev} = 0{,}4378$, $\lambda_{NoE} = 1$, and $\lambda_{Aul} = 1{,}7127$. The new values of the quotients from Aul can be found by adding 0,5381 to the old values or they can be calculated as $(1{,}7127 \cdot \frac{p_{Aul,j}}{d_l})$. Once more we sort the quotients within each party in descending order. Table 13.10 presents the interesting parts of the sorted party lists:

Table 13.10

Party B

Place	Quotient	Value	Classification	Distance	Global no.
5	NoE.B. 2	8,009	1st Outgoing	0,650	
6	Rev.B. 2	7,665			
7	Aul.B. 3	7,647			
8	NoE.B. 3	7,603			
9	Aul.B. 4	7,359	1st Benchmark		
10	NoE.B. 4	7,316			
11	Rev.B. 3	7,260			

Party D

Place	Quotient	Value	Classification	Distance	Global no.
8	NoE.D. 2	7,742	2nd Outgoing	0,424	2nd
9	Rev.D. 6	7,613			
10	Rev.D. 7	7,459			
11	NoE.D. 3	7,337	1st Outgoing	0,012	1st
12	Rev.D. 8	7,325	1st Benchmark		
13	Aul.D. 2	7,318	2nd Benchmark		
14	Rev.D. 9	7,207			

Party G

Place	Quotient	Value	Classification	Distance	Global no.
1	Rev.G. 1	8,327			
2	NoE.G. 1	7,916	1st Outgoing	0,688	
3	Aul.G. 1	7,675			
4	Rev.G. 2	7,634			
5	Rev.G. 3	7,228	1st Benchmark		

Sorting of quotients within the three parties after the second iteration + classification of quotients, distances between the xth outgoing and xth benchmark quotient, and global ranking of these distances when a downadjustment of λ_{NoE} is considered.

From Table 13.10 we find the assignment after the second iteration:

Table 13.11

	Parties					Representation		
	B	D	G	Total	Constraint	Under	Over	ρ
Rev	2	7	2	11	12	1	–	
NoE	2	3	1	6	5	–	1	
Aul	3	1	1	5	5	0	0	
	7	11	4			1	1	2

Number of seats assigned to the constituencies within the three parties, malrepresentation, and measure of goodness with the multiplier set: $\lambda_{Rev} = 0{,}4378$, $\lambda_{NoE} = 1$, $\lambda_{Aul} = 1{,}7127$.

The measure of goodness is now as low as $\rho = 2$. Since $\rho > 0$ we have to carry out yet another iteration.

13.5 Last iteration

We do not go through the calculations which produced the data in Table 13.12. From the explanations in earlier sections it should be clear how they are determined from the sorted party lists in Table 13.10.

Table 13.12

Global distances for downadjustment of NoE

No.	Benchmark quotient	Distance	ρ – effect
1	Rev.D. 8	0,012	2
2	Aul.D. 2	0,424	–
			2

Global distances for upadjustment of Rev

No.	Benchmark quotient	Distance	ρ – effect
1	NoE.D. 3	0,012	2
2	Aul.B. 3	0,387	–
			2

Global distances, benchmark quotients, and ρ-effect for each of the two malrepresented constituencies.

Both ρ-effect and representation selection choose to downadjust λ_{NoE} because of the rule regarding alternation of up and downadjustments. $\phi_{Rev} = 2$, so the upadjustment of λ_{Rev} would also reduce the measure of goodness to zero. The adjustment distance for NoE is $\chi_{NoE} = 0{,}3 \cdot 0{,}012 + (1 - 0{,}3) \cdot 0{,}424 \approx 0{,}3004$, which results in a new multiplier value of $\lambda_{NoE} = 1 \cdot e^{-0{,}3004} \approx 0{,}7405$. After recalculation of quotients from NoE, we get the following sorted party lists:

Table 13.13

Party B

Place	Quotient	Value
1	Aul.B. 1	8,745
2	NoE.B. 1	8,402
3	Rev.B. 1	8,358
4	Aul.B. 2	8,052
5	NoE.B. 2	7,708
6	Rev.B. 2	7,665
7	Aul.B. 3	7,647

Party D

Place	Quotient	Value
1	Rev.D. 1	9,404
2	Rev.D. 2	8,711
3	Rev.D. 3	8,306
4	NoE.D. 1	8,135
5	Rev.D. 4	8,018
6	Aul.D. 1	8,011
7	Rev.D. 5	7,795
8	Rev.D. 6	7,613
9	Rev.D. 7	7,459
10	NoE.D. 2	7,442
11	Rev.D. 8	7,325

Party G

Place	Quotient	Value
1	Rev.G. 1	8,327
2	Aul.G. 1	7,675
3	Rev.G. 2	7,634
4	NoE.G. 1	7,616

Sorting of assigned quotients within the three parties after the third iteration.

These party lists imply the following assignment:

Table 13.14

	Parties					Representation		
	B	D	G	Total	Constraint	Under	Over	ρ
Rev	2	8	2	12	12	0	0	
NoE	2	2	1	5	5	0	0	
Aul	3	1	1	5	5	0	0	
	7	11	4			0	0	0

Number of seats assigned to the constituencies within the three parties with the suitable multiplier set: $\lambda_{Rev} = 0{,}4378$, $\lambda_{NoE} = 0{,}7405$, $\lambda_{Aul} = 1{,}7127$.

ρ is zero, which means that the assignment in the left part of Table 13.14 is the optimal apportionment. Since the multipliers $\lambda_{Rev} = 0{,}4378$, $\lambda_{NoE} = 0{,}7405$, and $\lambda_{Aul} = 1{,}7127$ solve the problem, they form a suitable multiplier set.

Chapter 14: Empirical tests of the algorithm

In this chapter we test the behaviour of the apportionment algorithm empirically. The first three sections describe the framework for the tests. Section 14.1 describes the data sets utilized, section 14.2 follows up with a presentation of the five test parameters, while the third section presents the test method employed. To find out how different levels of the parameters influence the speed of the solution process, we carry out tests of the parameters. We test selection methods in section 14.4, relaxations in section 14.5, and initialization procedures in section 14.7. The main criterion in these tests is the average number of iterations. Section 14.6 presents an alternative way of measuring the speed of the algorithm, while we in the final section study the impact of bound vectors and matrix apportionment methods. The complete results of the tests carried out in this chapter can be found in Appendix 3.

14.1 Data sets

We have utilized data from general elections in 8 countries in the algorithm tests. The number of data sets from each country spans from 4 to 28 as shown in Table 14.1:

Table 14.1

Country	Sweden	Finland	Denmark	Norway	Iceland	(W.) Germany	Luxembourg	Austria	Total
Data sets	28	17	13	12	11	14	5	4	104

Number of data sets from each country.

Lists of the utilized data sets giving information about country, election year, etc are presented in Appendix 3. Be aware that the values of h, m, and n given in these lists are the actual values after the manipulations described below. The election data of interest to us are the vote matrices in absolute figures and the actual apportionments to constituencies and parties. Not all parties are of interest, a party must have won a seat in the election or received at least 1% of the votes at the national level to be included in our data sets. Every constituency has been included in the data sets. The exceptions to this rule are the Danish and Finnish data sets where a few small constituencies with special status have been excluded. Other deviations from the data set norm described above are presented in the paragraph below. These deviations concern only a few of the data sets.

We start with matters concerning parties. Some small parties which met our party criterion, but where only vote totals were available, have been omitted. The second party related matter concerns small parties in some West German data sets. Only percentage vote data given with one decimal's accuracy were available for these parties, so the deduced absolute vote figures are somewhat inaccurate estimates of the actual vote figures. Another matter regarding inaccuracy, the absolute vote figures from West Germany and Germany after World War 2 are given in multiples of a hundred. Now to the data set for the Swedish election in 1994. We have only got hold of vote data for 20 constituencies plus 4 quasi-constituencies for this election, so we operate with 24 constituencies, 4 less than the actual number. The last data set we comment on is for the Danish election in 1990. We have only got hold of percentage vote data given with one decimal's accuracy for this election. The vote matrix in absolute figures deduced from these data is therefore an inaccurate estimate of the actual vote matrix.

14.2 Test parameters

There are 5 test parameters; selection method, relaxation, initialization procedure, bound vector, and matrix apportionment method. We operate with 2 selection methods, 2 relaxations, 3 initialization procedures, 3 bound vectors, and 4 matrix apportionment methods. The apportionment algorithm is therefore run $2 \cdot 2 \cdot 3 \cdot 3 \cdot 4 = 144$ times for each data set. We record the initial measure of goodness and the number of iterations for each parameter combination, i.e. for each run of the algorithm. Be aware that all 5 parameters influence the number of iterations while the initial measure of goodness is unaffected by the selection method employed. The first three parameters are directly connected to the effectiveness of the algorithm, while the last two in tandem determine the optimal apportionment. We call them algorithm parameters and political parameters respectively. Bound vector and matrix apportionment method are called political parameters because they should be determined politically. Hence, the number of iterations should not influence the choice of bound vector and matrix apportionment method. The arguments put forward in sections 8.1 and 8.3 are relevant for the choice of bound vector, while the measurement of matrix bias in chapter 16 should be taken into consideration when deciding which matrix apportionment method to apply. Below we present the bound vectors and matrix apportionment methods employed in the tests.

Since we deal with equality constrained problems, the bound vector can be written as $\sigma = (\mathbf{R}, \mathbf{C}, h)$. For a given house size h, it is the combination of constituency and party bounds which distinguishes a bound vector from another. Some places we use the notation X-Y for a **bound vector**, where X and Y symbolize the apportionment methods used to determine the constituency bounds **R** and party bounds **C** respectively. We use the abbreviation **EL** for the actual apportionment, i.e. the apportionment determined by election law. One of the bound vectors we utilize is EL-EL, i.e. both constituency and party bounds are as determined in the election. The bounds are often far from proportional to the vote totals with EL-EL. This is one of the reasons for bringing in two

proportional bound vectors. We have chosen MF-MF and SD-HA, where the bounds are determined by applying the specified apportionment method on the vote totals. Based on the vector bias results from section 5.8 one might say that MF-MF gives unbiased bounds while SD-HA gives bounds which favour small constituencies and large parties.

The four matrix apportionment methods we use in the tests are $CP_{0,01}$, DM ($CP_{\frac{1}{3}}$), MF ($CP_{0,5}$), and HA (CP_1). $CP_{0,01}$ has about the same effect as SD (CP_0) and is used instead of SD because it is better suited to our Pascal program of the apportionment algorithm. MF and HA are natural choices, while DM is included based on an intuition that a constant parametric divisor method somewhere between SD and MF could be interesting for matrix apportionment purposes.

Since we operate with 3 bound vectors and 4 matrix apportionment methods, each data set may lead to $3 \cdot 4 = 12$ different apportionments. A data set results in fewer than 12 different apportionments when two of the bound vectors or all three are identical, or when different matrix apportionment methods result in the same matrix apportionment for a given bound vector. There are few instances of identical apportionments for our 104 data sets.

The algorithm parameters we use in the tests have been described in earlier chapters; the 2 relaxations in section 11.1, the 3 initialization procedures in section 11.3, and the 2 selection methods in sections 12.1 and 12.5. Furthermore, the utilized values for target weight and adjustment weight, $\tau = 0,3$ and $\upsilon = 0,3$ respectively, are results of the tests in sections 11.5 and 12.4 respectively.

Relaxation together with initialization procedure determine the starting position of the apportionment algorithm, while the selection method determines the path followed to the optimal apportionment. Thus, each apportionment is usually found through $2 \cdot 3 \cdot 2 = 12$ different paths. We are now ready to choose test method:

14.3 Test method

Our aim is to make the apportionment algorithm as efficient as possible given the political parameters, i.e. to solve matrix apportionment problems in as few iterations as possible. To find the best set-up of the algorithm, we carry out three separate tests. We test the two selection methods against each other in section 14.4, the two relaxations against each other in section 14.5, and the three initialization procedures against each other in section 14.7. For each level of a parameter we calculate the average number of iterations and sometimes also the average initial measure of goodness. These averages are calculated over all combinations of the four other parameters, i.e. over $2 \cdot 3 \cdot 3 \cdot 4 = 72$ cases in the selection method and relaxation tests and over $2 \cdot 2 \cdot 3 \cdot 4 = 48$ cases in the initialization procedure test. We calculate such averages for every data set.

To test a level of a parameter against another level of the same parameter, we compare the averages for the two levels. We let φ denote the ratio and η the difference between two such averages. This notation is used for both iteration and $\mathring{\rho}$ averages. We assume that ratios from different data sets are independent of each other. Moreover, we assume that they follow a normal distribution which mean and variance both are unknown. Similar assumptions are employed for differences. Our estimates of the means are denoted $\bar{\varphi}$ and $\bar{\eta}$ respectively. They are calculated as follows:

$$(14.1) \qquad \bar{\varphi} = \frac{1}{n} \cdot \sum_{a=1}^{n} \varphi_a \qquad\qquad \bar{\eta} = \frac{1}{n} \cdot \sum_{a=1}^{n} \eta_a$$

where a is the index for data sets and n is the number of data sets (observations). The variances are estimated the following way:

$$(14.2) \qquad s_{\varphi}^2 = \frac{1}{n-1} \cdot \sum_{a=1}^{n} (\varphi_a - \bar{\varphi})^2 \qquad s_{\eta}^2 = \frac{1}{n-1} \cdot \sum_{a=1}^{n} (\eta_a - \bar{\eta})^2$$

The estimation process is carried out for each country. We also calculate "universal" estimates based on all 104 data sets available. Below we present the null hypotheses and test statistics we employ in the tests. The null hypothesis for both ratios and differences is that the averages are equal, which means:

(14.3) $H_0: \varphi = 1$ $H_0: \eta = 0$

The test statistics employed for ratios and differences are:

(14.4) $$T_\varphi = \frac{\bar{\varphi} - 1}{\frac{s_\varphi}{\sqrt{n}}} \qquad T_\eta = \frac{\bar{\eta} - 0}{\frac{s_\eta}{\sqrt{n}}}$$

Given the assumptions above, both T_φ and T_η are t-distributed with n - 1 degrees of freedom. We use a level of significance of 5% in the tests.

In this paragraph we consider possible violations of the assumptions above. There is reason to believe that some of the ratios and differences in our tests are affected by the size of the vote matrices $m \times n$. For most countries the number of constituencies m and parties n are quite stable over time. Thus, the assumption that observations are drawn from the same normal distribution should usually be satisfied for individual countries. Sweden and Germany are notable exceptions regarding size stability. The Swedish data sets prior to 1921 have twice as many constituencies as the later sets, while the German data sets have a quite different size than the West German sets. For these countries we have also carried out the tests with the anomalous data sets removed. The resultant estimates did not differ that much from the estimates presented in the following sections. As mentioned earlier, we calculate universal estimates based on all 104 data sets available. Due to large differences in vote matrix sizes among countries, the assumption that all observations are drawn from the same normal distribution is presumably violated in this case. If the actual probability distribution has thicker tails than Student's t-distribution, the reported P-values are too low. This should be kept in mind when studying the universal estimates in the following sections.

We end this section by suggesting an alternative way of testing the apportionment algorithm. In hindsight, it would have been an idea to use a five factor model, see [D&C] (page 145→), to analyse the performance of the algorithm. This way we would have got estimates of the impact of each parameter level plus estimates of interaction effects between different parameters. The best combination of relaxation, initialization procedure, and selection method found with such a test would presumably be identical to the combination of the best relaxation, the best initialization procedure, and the best selection method found with our tests. The reason for this postulate is the clear winners in each of our three tests.

14.4 Comparison of selection methods

In this section we compare the two selection methods. The basis for this comparison is the average number of iterations for each data set. Let us denote the average with ρ-effect selection (O) and the average with representation selection (R). A possibility is to look at the difference between these averages, i.e. (O) - (R). However, use of (O) - (R) would have meant that we expected the same difference for a data set where the average initial measure of goodness $\mathring{\rho}$ is high as for a data set where the average $\mathring{\rho}$ is low. Since ρ-effect selection makes use of more information than representation selection before it decides which multiplier to adjust, we expect it to perform somewhat better than the latter in every iteration of the algorithm. Then it is natural to calculate the iteration ratio $\frac{(O)}{(R)} = \varphi$. The null hypothesis is that there is no difference in performance between the two selection methods, i.e. $\varphi = 1$, while our alternative hypothesis is that ρ-effect selection is better:

(14.5) $\qquad H_A: \varphi < 1$

Thus, we employ a one-sided test. We carry out the test as explained in section 14.3. The estimated iteration ratios $\bar{\varphi}$ and the statistical significance of these estimates are presented in Table 14.2:

Table 14.2

Country	Sweden	Finland	Denmark	Norway	Iceland	(W.) Germany	Luxembourg	Austria	All
Estimated (O)/(R) ratio	0,91 **	0,87 **	0,88 **	0,89 **	0,83 **	0,80 **	0,79 **	0,72 *	0,86 **

Estimated iteration ratios $\bar{\varphi} = \frac{(O)}{(R)}$ for selection methods.

We see that all estimates but one are significant at the 1%-level. Moreover, the estimate for the last country, Austria, is significant at the 5%-level. Of the individual data sets, there was only one Swedish set where $\varphi > 1$, i.e. where the average number of iterations with representation selection was lower than with ρ-effect selection. The conclusion is crystal clear, ρ-effect selection reduces the number of iterations compared with representation selection. For the collection of all data sets ρ-effect selection used on average $\frac{1 - 0,86}{1}$ = 14% less iterations than representation selection. Whether this reduction is big enough to make up for the increased workload with ρ-effect selection is another question. The estimates in Table 14.2 hide the fact that compared with representation selection, ρ-effect selection is more efficient working on party relaxed than on constituency relaxed problems. For the collection of all data sets the estimated iteration ratios with these two relaxations were 0,79 and 0,90 respectively.

14.5 Comparison of relaxations

As mentioned in [H&J] (page 21), constituency relaxation may be more acceptable politically than party relaxation. Let us forget the political aspect for the moment and see how the relaxations fare. Below we present two ways of comparing them. Since we have not got any basis for saying that one of the relaxations is better than the other, we employ two-sided tests.

The first comparison is based on the initial measure of goodness $\mathring{\rho}$. A low $\mathring{\rho}$ is normally better than a higher one. For each data set we calculate the average $\mathring{\rho}$ with constituency relaxation, denoted [C], and with party relaxation, denoted [P]. We calculate the difference between these two averages [P] - [C] = η to compare the relaxations. Since this means that we expect the same difference for every

data set, it might have been better to use the $\mathring{\rho}$-ratio $\frac{[C]}{[P]}$. The null hypothesis is that the relaxations are equally good, i.e. $\eta = 0$, while our alternative hypothesis is that one of them is better:

(14.6) $H_A: \eta \neq 0$

The estimated $\mathring{\rho}$-differences and the statistical significance of these estimates are presented in Table 14.3:

Table 14.3

Country	Sweden	Finland	Denmark	Norway	Iceland	(W.) Germany	Luxembourg	Austria	All
Estimated [P] - [C]	4,46 **	4,67 **	8,67 **	4,45 **	-1,83 **	1,63	- 0,07	1,75	3,65 **

Estimated $\mathring{\rho}$-differences $\bar{\eta}$ = [P] - [C] for relaxations.

The estimate for the collection of all data sets is significant at the 1%-level. On average, party relaxation results in an average $\mathring{\rho}$ which is 3,65 higher than with constituency relaxation. This is the opposite result of what we experienced in the test in section 11.5. An explanation is that $\mathring{\rho}$ there was based on apportionment initialization, while $\mathring{\rho}$ here is based on all three initialization procedures. Let us return to the figures in Table 14.3. The estimates for the Nordic countries are significant at the 1%-level, while the null hypothesis is not rejected for the other three countries. Notice that the estimate for Iceland has the opposite sign of the other significant estimates. In fact, party relaxation resulted in the lowest average $\mathring{\rho}$ for all 11 Icelandic data sets. If we look at all 104 data sets utilized, 76 had their lowest average $\mathring{\rho}$ with constituency relaxation, 27 with party relaxation, and for 1 set the averages were equal.

The drawback of basing the comparison on the initial measure of goodness $\mathring{\rho}$ is that $\mathring{\rho}$ only tells something about the starting position of the apportionment algorithm. $\mathring{\rho}$ does not tell us how accommodating the relaxation is towards the execution of the algorithm. To cover this aspect we utilize the number of iterations once again. From the figures in Table 14.3 one should possibly expect party relaxation to require a higher number of iterations than constituency relaxation. Whether this is the case is tested below.

We let (C) and (P) denote the average number of iterations for a data set with constituency and party relaxation respectively. Above we found that the average $\mathring{\rho}$ was lower with constituency than with party relaxation. A possibility is to assume that this difference combined with the workings of the apportionment algorithm lead to a difference in the number of iterations (P) - (C). However, since we expect (P) - (C) to vary with the size of the problems, we use an iteration ratio. We have chosen the ratio $\frac{(P)}{(C)} = \varphi$. The null hypothesis is that the two relaxations are equally good, i.e. $\varphi = 1$, while our alternative hypothesis is that one of them is better:

(14.7) $\qquad H_A: \varphi \neq 1$

Table 14.4 presents the estimated iteration ratios $\bar{\varphi}$ for relaxations and the statistical significance of these estimates:

Table 14.4

Country	Sweden	Finland	Denmark	Norway	Iceland	(W.) Germany	Luxembourg	Austria	All
Estimated (P)/(C) ratio	0,32 **	0,65 **	0,70 **	0,51 **	0,73 **	0,54 **	1,37	0,67	0,58 **

Estimated iteration ratios $\bar{\varphi} = \frac{(P)}{(C)}$ for relaxations.

Contrary to what the $\mathring{\rho}$-differences suggested, party relaxation is best. On average, it only uses 58% of the iterations constituency relaxation does. This estimate is significant at the 1%-level. With two exceptions, the estimates for the countries are significant at the 1%-level in favour of party relaxation. Luxembourg is the only country where constituency relaxation seems preferable, but the estimate is not significant. Of the iteration ratios for individual data sets, $\varphi > 1$ for 1 German, 1 Austrian, and all but one of the Luxembourg data sets. A majority of these data sets are characterized by having more or as many parties as there are constituencies, i.e. $n \geq m$. An explanation of why it is usually preferable to relax the index with fewest constraints follows in the next section.

14.6 The average decrease in ρ per iteration

In this section we present a measure which both takes the difficulty of the task, as measured by the initial measure of goodness $\mathring{\rho}$, and the speed of the solution process, as measured by the number of iterations, into account. The measure, denoted ι, is given by (14.8). We call it the **average decrease in ρ per iteration.**

(14.8) $$\iota = \frac{\text{Initial measure of goodness}}{\text{Number of iterations}}$$

ι was constructed to compare the performance of the two selection methods. It is well suited for this task because $\mathring{\rho}$ is the same for both. For each selection method we find the average ι for a data set by calculating ι for every case where $\mathring{\rho} > 0$ and taking the average. Notice that the average ι is not generally equal to the ratio $\frac{\text{Total initial measure of goodness}}{\text{Total number of iterations}}$. The average ι for each combination of relaxation, selection method, and data set can be found in Appendix 3. Moreover, the results of the selection method test based on ι are also presented there. These results are similar to the ones in section 14.2.

Our original hypothesis regarding ι was that it, in broad terms, would be an increasing function of $\mathring{\rho}$. To check this hypothesis, we drew graphs, one for each combination of country and relaxation. For each combination of data set and relaxation we calculated the average $\mathring{\rho}$ and ι and plotted the point $(\mathring{\rho},\iota)$ in the relevant graph. In most of the graphs ι was fairly constant. Thus, ι seems to be independent of $\mathring{\rho}$, at least for the averages. However, the average ι differed from one graph to the next, so let us look at the average ιs for the different combinations of country and relaxation:

Table 14.5

Country	Constituency relaxation			Party relaxation		
	Initial ρ	Selection		Initial ρ	Selection	
		Representation	ρ - effect		Representation	ρ - effect
Austria	7,02	1,08	1,33	8,76	2,08	3,17
Denmark	19,37	1,02	1,13	28,04	1,88	2,28
Finland	10,82	0,93	1,05	15,48	1,75	2,26
(W.) Germany	10,20	1,13	1,36	11,83	2,58	3,28
Iceland	8,44	1,37	1,59	6,61	1,75	2,08
Luxembourg	3,67	1,42	1,76	3,59	1,44	1,66
Norway	17,50	0,95	1,05	21,94	2,27	2,71
Sweden	19,01	0,88	0,96	23,48	3,05	3,81

Average $\mathring{\rho}$ and ι s for the different combinations of country and relaxation.

ι gives important information about the speed of the solution process. Let us illustrate how the ι data in Table 14.5 can be used to estimate the length of the solution process for a new problem: Assume that we face a constituency relaxed Austrian apportionment problem where $\mathring{\rho} = 4$. Table 14.5 tells us that the solution process should take about $\frac{4}{1,33} \approx 3$ iterations if ρ-effect selection is employed.

What we find most interesting with the figures in Table 14.5 is that ι is much higher with party than with constituency relaxation. Here follows an explanation for this: Because $m > n$ in almost all our data sets, there are often more malrepresented constituencies than parties for a given value of $\mathring{\rho}$. The average malrepresentation per malrepresented item is higher with party than with constituency relaxation in such situations. In many cases this leads to a higher maximal malrepresentation with party relaxation. The end result is often a higher ρ-effect and thereby a higher ι with party relaxation.

To get a somewhat different view of the relationship between the initial measure of goodness $\mathring{\rho}$ and the number of iterations than ι gives, we calculate correlation coefficients between the two. We distinguish between constituency and party relaxation in these calculations, which are based on the average $\mathring{\rho}$ and average number of iterations for each data set. The calculated correlation coefficients are presented in Table 14.6 below. With one exception, the coefficients are larger

than 0,55.

Table 14.6

		Relaxation	
Country	# of data sets	Constituency	Party
Austria	4	0,66	0,92
Denmark	13	0,70	0,56
Finland	17	0,65	0,88
(W.) Germany	14	0,96	0,94
Iceland	11	0,77	0,74
Luxembourg	55	0,95	0,80
Norway	12	0,67	0,19
Sweden	28	0,82	0,90
All countries	104	0,92	0,81

Correlation coefficients between average $\mathring{\rho}$ and average number of iterations.

14.7 Comparison of initialization procedures

The initialization process is about finding a good starting position for the apportionment algorithm. Since apportionment initialization makes use of more information about election situations than quota ratio initialization does and no initialization does not utilize any information at all, we expect apportionment initialization to be better than quota ratio initialization and quota ratio initialization to be better than no initialization. Our hypothesis is that a good initialization gives the algorithm a head start compared with a not so good initialization. This explains our use of differences in the tests below.

We start the comparison of the three initialization procedures by utilizing the initial measure of goodness $\mathring{\rho}$ again. Let [N], [Q], and [A] denote the average $\mathring{\rho}$ for a data set with no, quota ratio, and apportionment initialization respectively. We estimate the head start of an initialization procedure in relation to another procedure as the difference in average $\mathring{\rho}$ between the two. Estimates are calculated for the differences [N] - [Q] and [Q] - [A]. We use the notation η for both. The null hypothesis for each difference is that the two initialization

procedures in question are equally good, i.e. $\eta = 0$. Our alternative hypothesis is that the latter initialization procedure in a difference pair is best, i.e. has the lowest average $\mathring{\rho}$:

(14.9) $H_A: \eta > 0$

The results of these one-sided tests are presented in Table 14.7:

Table 14.7

Country	Sweden	Finland	Denmark	Norway	Iceland	(W.) Germany	Luxembourg	Austria	All
Estimated [N] - [Q]	2,40 **	2,18 **	1,58 **	4,63 **	3,22 **	4,81 **	1,87 **	1,40 **	2,86 **
Estimated [Q] - [A]	15,62 **	8,11 **	13,84 **	11,24 **	3,37 **	6,56 **	1,18 *	5,48 **	10,07 **

Estimated $\mathring{\rho}$-differences $\bar{\eta}$ = [N] - [Q] and $\bar{\eta}$ = [Q] - [A] for initialization procedures.

All estimates are positive and significant at the 1%-level, except [Q] versus [A] for Luxembourg which is significant at the 5%-level. From the figures in Table 14.7 we draw the conclusion that apportionment initialization is much better than quota ratio initialization, which itself is better than no initialization. Of the $\mathring{\rho}$-differences for individual data sets, only [Q] - [A] for the earliest Luxembourg set was negative. Finally, be aware that estimates for the difference [N] - [A] can be found by adding the estimates for [N] - [Q] and [Q] - [A] from Table 14.7.

As in section 14.5, the interesting point is how the starting position is transformed into number of iterations. Given the same value of (14.8) for every run of the apportionment algorithm, $\mathring{\rho}$ and the number of iterations would have told identical stories. This will not happen in practice. Let (N), (Q), and (A) denote the average number of iterations for a data set with no, quota ratio, and apportionment initialization respectively. To compare the initialization procedures we calculate the iteration differences (N) - (Q) and (Q) - (A). The null and alternative hypotheses regarding these differences are the same as for the $\mathring{\rho}$-differences above. Table 14.8 presents the estimated iteration differences and the statistical significance of these estimates given a one-sided test:

Table 14.8

Country	Sweden	Finland	Denmark	Norway	Iceland	(W.) Germany	Luxembourg	Austria	All
Estimated (N) - (Q)	1,43 **	0,83 **	0,61 *	1,49 **	1,06 **	0,49 **	0,74	0,31	1,00 **
Estimated (Q) - (A)	3,65 **	2,33 **	2,27 **	1,98 **	1,44 **	2,54 **	0,84 **	2,39 **	2,50 **

Estimated iteration differences $\bar{\eta}$ = (N) - (Q) and $\bar{\eta}$ = (Q) - (A) for initialization procedures.

We see that all estimates are positive. Moreover, all estimates for (Q) - (A) and most estimates for (N) - (Q) are significant at the 1%-level. The results in Table 14.8 confirm that apportionment initialization is best and quota ratio initialization second best. However, there were some data sets with a different ordering; (N) - (Q) was negative for 3 Swedish, 2 Finnish, 3 Danish, 2 German, and 1 Austrian set, while (Q) - (A) was negative for 1 Danish set. A point worth noticing when studying Table 14.7 and Table 14.8 is that overall $\mathring{\rho}$-differences of 2,86 and 10,07 lead to overall iteration differences of 1,00 and 2,50 respectively.

Another figure which tells something about an initialization procedure is the number of cases solved directly. The initialization procedures were able to solve the following number of cases:

Table 14.9

Initialization procedure	No	Quota ratio	Apportionment	Total
Direct solutions	14	20	90	124

Number of direct solutions with the three initialization procedures.

We observe that the ranking of the three initialization procedures is the same as with the tests above. The countries with most direct solutions were Luxembourg with 42, West Germany with 40, and Iceland with 26. When evaluating the figures in Table 14.9, it should be taken into account that a total of 2496 initial multiplier sets have been created by each initialization procedure; $2 \cdot 3 \cdot 4$ sets for each of the 104 data sets. Hence, the solution percentages are low; 3,6% with apportionment initialization and below 1,0% with the two other initialization procedures.

14.8 Final remarks

Based on the results of the tests in earlier sections, we draw the following conclusion: The best way to solve a matrix apportionment problem in as few iterations as possible is to use party relaxation, apportionment initialization, and ρ-effect selection.

As pointed out in section 14.2, the choice of bound vector and matrix apportionment method should not be ruled by iteration figures. Nevertheless, it is interesting to see how different levels of these two parameters affect $\mathring{\rho}$ and the number of iterations:

First, we look at the effect of the three bound vectors. Below we let E, M, and S stand for the bound vectors EL-EL, MF-MF, and SD-HA respectively. It would have been possible to test the bound vectors against each other using the same kind of tests as in earlier sections, but we restrict ourselves to ordinal values here. We use the average initial measure of goodness $\mathring{\rho}$ and average number of iterations for each country to rank the bound vectors. X > Y means that bound vector X is better than bound vector Y, i.e. that X has a lower average. We get the following rankings for the different countries:

Table 14.10

Country	Sweden	Finland	Denmark	Norway	Iceland	(W.) Germany	Luxembourg	Austria
Average $\mathring{\rho}$	M>E>S	M>S>E	E>M>S	M>S>E	M>S>E	M>S>E	M>S>E	S>M>E
Average iterations	E>S>M	M>E>S	E>S>M	E>S>M	M>S>E	M>E>S	M>E>S	S>E>M

Rankings of the three bound vectors based on average $\mathring{\rho}$ and average number of iterations.

Based on pairwise comparisons, the overall ranking is M > S > E for $\mathring{\rho}$, while we run into the Condorcet paradox for number of iterations because E = M, E > S, but M = S. Thus, there does not seem to be much difference between the bound vectors iterationwise, although E results in the lowest sum over all countries.

Next on the agenda is a study of the effect of the four matrix apportionment methods. We use the same methodology as for bound vectors to rank these methods. Table 14.11 below shows the rankings. In the table, C, D, M, and H stand for $CP_{0,01}$, DM, MF, and HA respectively.

Table 14.11

Country	Sweden	Finland	Denmark	Norway	Iceland	(W.) Germany	Luxembourg	Austria
Average ρ	M>D>H>C	M>D>H>C	D>M>H>C	M>D>H>C	D>M>C>H	M>D>H>C	M>D>H>C	M>D>H>C
Average iterations	M>D>H>C	M>D>C>H	D>M>H>C	M>D>H>C	M>D>H>C	M>D>C>H	M>D>H>C	M>D>H>C

Rankings of the four matrix apportionment methods based on average $\mathring{\rho}$ and average number of iterations.

The picture is much clearer here than it was for the bound vectors. The overall ranking is MF > DM > HA > $CP_{0,01}$ for both criteria. Furthermore, notice that MF and DM never are ranked below second place in Table 14.11.

$\mathring{\rho}$ and iteration data for each data set are presented in Appendix 3. A look at the iteration figures strengthens the impression that $CP_{0,01}$ is most troublesome iterationwise. Of the four matrix apportionment methods, MF results in the lowest number of iterations overall, although DM is not far behind. The initial measure of goodness $\mathring{\rho}$ is much higher with $CP_{0,01}$ and HA than with DM and MF. This indicates that the three initialization procedures have problems handling constant parametric divisor methods with t in the neighbourhood of 0 or 1. Whether any better initialization procedure exists for such divisor methods is an unanswered question, at least in this dissertation.

Chapter 15: Controlled rounding

Apportionment of seats in elected assemblies is not the only application of divisor methods. They can also be used for problems which presently are solved by controlled rounding. Controlled rounding has several applications, for a review of these we refer to [C&C&E] (page 904-905). Our focus in this chapter is on controlled rounding as an apportionment method. We start by formulating the two-dimensional controlled rounding problem. A divisor method formulation in the same spirit follows in the second section. In section 15.3 we present a formulation which restricts the rounding to internal entries. Section 15.4 looks at the pros and cons of rounding all entries compared to rounding only internal entries. The final section discusses briefly whether divisor methods are suitable for traditional controlled rounding problems.

15.1 Two-dimensional formulation

Controlled rounding is the problem of rounding all entries in a one-, two-, or multi-dimensional matrix **b** to integer multiples of a positive integer base b subject to the requirements:

Additivity: The sum of the rounded values along any row or column equals the rounded total value of the corresponding row or column.

Adjacency: Each entry in **b** is rounded to an adjacent integer multiple of b.

Below follows an introduction to the two-dimensional controlled rounding problem. The extension to higher dimensions is straightforward. However, some multi-dimensional problems are infeasible. The formulation below includes sum entries while the formulation in section 15.3 does not.

We start by introducing some new notation. The sets $M^S = \{1, 2, ..., m, (m + 1)\}$ and $N^S = \{1, 2, ..., n, (n + 1)\}$ are related to the rows and columns respectively. In this section we let **b** be a $(m + 1) \times (n + 1)$ matrix with real valued entries b_{ij}, where $i \in M^S$ and $j \in N^S$. We call the entries in the m x n submatrix in the upper left corner of **b** **internal entries**. The bottom row, i.e. row $(m + 1)$, contains the column sums b_{Mj} and the rightmost column, i.e. column $(n + 1)$ contains the row sums b_{iN}. These entries are referred to as **total entries**. Finally, the entry in the lower right corner, which is the sum of all internal entries b_{MN}, is called **grand total entry**.

We let **a** denote the controlled rounding of **b.** All entries in **a** are non-negative integers, i.e. $a_{ij} \in N_0$ for all $i \in M^S$ and $j \in N^S$. All entries in **b** shall be rounded to the rounding base b, where b is a positive integer. The set of all non-negative multiples of b, which we denote $b \cdot N_0$, is important in the following. We round an entry to one of the adjacent integer multiples of b, i.e. either down to $\lfloor \frac{b_{ij}}{b} \rfloor$ or up to $\lceil \frac{b_{ij}}{b} \rceil$. These roundings satisfy the **adjacency** condition, which can be written as:

$$\lfloor \frac{b_{ij}}{b} \rfloor \leq a_{ij} \leq \lceil \frac{b_{ij}}{b} \rceil \qquad \forall\ i \in M^S, j \in N^S \tag{15.1}$$

$\lfloor \frac{b_{ij}}{b} \rfloor$ and $\lceil \frac{b_{ij}}{b} \rceil$ are equal if and only if b_{ij} belongs to the set $b \cdot N_0$. A rounding scheme which guarantees that whenever $b_{ij} \in b \cdot N_0$ then $a_{ij} = \frac{b_{ij}}{b}$, i.e. that all whole multiples of b always result in a corresponding number of entities, is called **zero restricted**. We observe that adjacency as defined by (15.1) implies zero restrictedness.

The optimal controlled rounding **a** minimizes the l_p-norm, [C&E] (page 425):

$$(15.2) \qquad l_p = l_p(\mathbf{a},\mathbf{b}) = \Big[\sum_{i\in M^S}\sum_{j\in N^S} |b\cdot a_{ij} - b_{ij}|^p\Big]^{1/p} \qquad \text{for } 1 \le p < \infty$$

$$(\text{or } l_\infty = \lim_{p\to\infty} l_p \quad \text{for } p = \infty)$$

subject to the additivity and adjacency requirements.

A controlled rounding problem can be standardized by carrying out two simple transformations. In the first transformation each entry in **b** is divided by the base b, which yields an equivalent controlled rounding problem with base equal to 1. To relate the description of controlled rounding to the apportionment problem, we let **b** be a vote matrix and allow b to be a positive rational number. Then the transformation above with $b = \frac{p_{MN}}{h}$ is similar to calculating quotas.

The zero restrictedness condition applied to base 1 problems demands that all integers shall result in a corresponding number of entities. In the apportionment problem, the grand total entry is equal to h and therefore integer. However, all other entries are usually non-integer. With respect to the rounding process we can ignore the integers and focus on the fractions. We denote the integer part of an entry $\bar{a}_{ij} = \lfloor \frac{b_{ij}}{b} \rfloor$ and the fractional part $\tilde{b}_{ij} = \frac{b_{ij}}{b} - \bar{a}_{ij}$. Thus, the matrix $\tilde{\mathbf{b}}$ consists entirely of entries smaller than 1. In the second standardization transformation integers are subtracted from the internal entries. The resultant matrix is denoted $\hat{\mathbf{b}}$. In this matrix each original internal entry b_{ij} is replaced by $\hat{b}_{ij} = \frac{b_{ij}}{b} - \bar{a}_{ij} = \tilde{b}_{ij}$. Thus, $0 \le \hat{b}_{ij} < 1$ for all $i \in M$ and $j \in N$. The total entries in $\hat{\mathbf{b}}$ are found as the sums of the transformed internal entries:

$$(15.3) \qquad \hat{b}_{i(n+1)} = \sum_{j\in N} \hat{b}_{ij} \qquad \forall\ i \in M$$

$$(15.4) \qquad \hat{b}_{(m+1)j} = \sum_{i\in M} \hat{b}_{ij} \qquad \forall\ j \in N$$

$$(15.5) \qquad \hat{b}_{(m+1)(n+1)} = \sum_{i\in M}\sum_{j\in N} \hat{b}_{ij}$$

These total entries may well be larger than 1. We follow up the second transformation by introducing the 0/1-variable $\hat{a}_{ij}$. $\hat{a}_{ij}$ is equal to 0 when $\tilde{b}_{ij}$ is rounded down and equal to 1 when $\tilde{b}_{ij}$ is rounded up. Each entry in **a** can therefore be expressed as $a_{ij} = \bar{a}_{ij} + \hat{a}_{ij}$. After the two standardization transformations, the controlled rounding problem is on **canonical form**. For a problem on canonical form we may write the adjacency condition as:

$$|\hat{a}_{ij} - \tilde{b}_{ij}| < 1 \qquad \forall\ i \in M^S, j \in N^S \tag{15.6}$$

Notice the resemblance between (15.6) and equation (3.3), which was one way of formulating staying within the quota. We therefore conclude that the adjacency condition is the same as staying within the quota transferred to a two-dimensional setting.

To formulate the two-dimensional controlled rounding problem on canonical form as a constrained optimization problem, we convert (15.2) to the objective function $z_p = (l_p(\mathbf{a},\mathbf{b}))^p$:

$$\min z_p = \sum_{i \in M^S} \sum_{j \in N^S} |\hat{a}_{ij} - \tilde{b}_{ij}|^p \qquad \text{for } 1 \le p < \infty \tag{15.7}$$

The corresponding objective function for the one-dimensional controlled rounding problem: $\min \sum_{i \in M^S} |\hat{a}_i - \tilde{b}_i|^p$ is equivalent to using LF, see [H-A] (page 10). This fact together with the connection between staying within the quota and adjacency established above are the reasons why we interpret two-dimensional controlled rounding as LF in two dimensions.

[C&E] (page 428) deduce the following formula for z_p:

$$z_p = \sum_{i \in M^S} \sum_{j \in N^S} ([(1 - \tilde{b}_{ij})^p - (\tilde{b}_{ij})^p] \cdot \hat{a}_{ij}) + \sum_{i \in M^S} \sum_{j \in N^S} (\tilde{b}_{ij})^p \tag{15.8}$$

z_p is the function which shall be minimized. It is linear in the variables $\hat{a}_{ij}$. z_p's last term is independent of $\hat{a}_{ij}$, i.e. a constant, and may therefore be omitted from the objective function. Furthermore, it can be verified that z_2 is equal to z_1 plus a

constant, so the optimal controlled roundings of **b** with respect to l_1 and l_2 coincide. We are now ready to present a constrained optimization formulation of the two-dimensional controlled rounding problem on canonical form:

$$(15.9) \qquad \min \sum_{i \in M^S} \sum_{j \in N^S} \left([(1 - \tilde{b}_{ij})^p - (\tilde{b}_{ij})^p] \cdot \hat{a}_{ij} \right) \qquad \text{for } 1 \le p < \infty$$

subject to the constraints

$$(15.10) \qquad \sum_{j \in N} \hat{a}_{ij} + (1 - \hat{a}_{i(n+1)}) = \lfloor \hat{b}_{i(n+1)} + 1 \rfloor \qquad \forall\ i \in M$$

$$(15.11) \qquad \sum_{i \in M} \hat{a}_{ij} + (1 - \hat{a}_{(m+1)j}) = \lfloor \hat{b}_{(m+1)j} + 1 \rfloor \qquad \forall\ j \in N$$

$$(15.12) \qquad \sum_{j \in N} (1 - \hat{a}_{(m+1)j}) + \hat{a}_{(m+1)(n+1)} = \sum_{j \in N} \lfloor \hat{b}_{(m+1)j} + 1 \rfloor - \lfloor \hat{b}_{(m+1)(n+1)} \rfloor$$

$$(15.13) \qquad \sum_{i \in M} (1 - \hat{a}_{i(n+1)}) + \hat{a}_{(m+1)(n+1)} = \sum_{i \in M} \lfloor \hat{b}_{i(n+1)} + 1 \rfloor - \lfloor \hat{b}_{(m+1)(n+1)} \rfloor$$

$$(15.14) \qquad \hat{a}_{ij} \ge 0 \qquad \forall\ i \in M^S, j \in N^S$$

$$(15.15) \qquad \hat{a}_{ij} \le 1 \qquad \forall\ i \in M^S, j \in N^S$$

The additivity and adjacency requirements are taken care of by (15.10) - (15.13) and (15.14) - (15.15) respectively. Since the problem formulated above is a two-dimensional capacitated transportation problem where all bounds are integer valued, every basic solution will be integer valued too. Now Theorem 15.1, proven by [C&E] (page 426-427), comes in handy:

Theorem 15.1
The two-dimensional controlled rounding problem on canonical form always has a feasible solution.

After the optimal solution $\hat{\mathbf{a}}$ has been determined, the optimal solution to the original controlled rounding problem is found as $\mathbf{a} = \bar{\mathbf{a}} + \hat{\mathbf{a}}$.

15.2 Divisor method formulation with total entries

The controlled rounding formulation above gives us an idea for a divisor method formulation with total entries. We build this formulation along the lines of the divisor method formulation presented at the end of section 10.1. What is different here is that **p** and **a** are $(m + 1) \times (n + 1)$ matrices because they include total entries. Another difference is the absence of predetermined constituency and party bounds. The reason for this is the simultaneous apportionment to total and internal entries. Now to the formulation:

(15.16) $$\max \sum_{(i,j)\in S} \sum_{l\in H} [\ln(\tfrac{p_{ij}}{d_l}) \cdot a_{ijl}]$$

subject to the constraints

(15.17) $$\sum_{j\in N} \sum_{l\in H} a_{ijl} = \sum_{l\in H} a_{i(n+1)l} \qquad \forall\ i \in M$$

(15.18) $$\sum_{i\in M} \sum_{l\in H} a_{ijl} = \sum_{l\in H} a_{(m+1)jl} \qquad \forall\ j \in N$$

(15.19) $$\sum_{j\in N} \sum_{l\in H} a_{(m+1)jl} = h$$

(15.20) $$\sum_{i\in M} \sum_{l\in H} a_{i(n+1)l} = h$$

(15.21) $$a_{ijl} \geq 0 \qquad \forall\ (i,j) \in S, l$$

(15.22) $$a_{ijl} \leq 1 \qquad \forall\ (i,j) \in S, l$$

(15.23) $$a_{ijl} = 0 \qquad \forall\ (i,j) \in \bar{S}, l$$

We end this section with a different topic. In the previous section two standardization transformations were introduced. The matter we will look into here is whether these transformations are applicable to problems which are going to be solved by a divisor method. The transformation to base 1 creates no difficulty since every divisor series can be multiplied by a common factor like b. As told in the previous section, $b = \frac{p_{MN}}{h}$ will transform a vote matrix to a quota matrix. However, the second transformation, namely subtraction of integers,

cannot be used. The reason is that none of the divisor methods satisfies the adjacency condition (15.1) in all situations. Situations which violate the adjacency condition can be constructed the same way as Example 3.2 in section 3.4. The consequence of determining the divisor apportionment from the canonical form of the problem is that this apportionment may deviate from the correct divisor apportionment, which is the one found from the original problem or the base 1 problem.

15.3 Rounding of only internal entries

Since controlled rounding can be viewed as an apportionment method, terms from the matrix apportionment problem are used interchangeably with terms from the controlled rounding problem in the remainder of this chapter. The controlled rounding formulation in section 15.1 included total entries. The consequence was that the constituency apportionment $a_{iN} = a_{i(n+1)}\ \forall\ i \in M$ and party apportionment $a_{Mj} = a_{(m+1)j}\ \forall\ j \in N$ were determined during the apportionment process. In this section we operate with predetermined constituency and party bounds, as we have done in earlier chapters. Thus, only the internal entries in **b** are rounded. The constrained optimization formulation of this controlled rounding problem on canonical form is presented below. In this formulation we let p = 1.

(15.24) $$\min z_1 = \sum_{i \in M} \sum_{j \in N} [(1 - 2 \cdot \tilde{b}_{ij}) \cdot \hat{a}_{ij}]$$

subject to the constraints

(15.25) $$r_i - \sum_{j \in N} \bar{a}_{ij} \leq \sum_{j \in N} \hat{a}_{ij} \leq R_i - \sum_{j \in N} \bar{a}_{ij} \qquad \forall\ i$$

(15.26) $$c_j - \sum_{i \in M} \bar{a}_{ij} \leq \sum_{i \in M} \hat{a}_{ij} \leq C_j - \sum_{i \in M} \bar{a}_{ij} \qquad \forall\ j$$

(15.27) $$\sum_{i \in M} \sum_{j \in N} \hat{a}_{ij} = h - \sum_{i \in M} \sum_{j \in N} \bar{a}_{ij}$$

(15.28) $$\hat{a}_{ij} \geq 0 \qquad \forall\ i,j$$

(15.29) $$\hat{a}_{ij} \leq 1 \qquad \forall\ i,j$$

After the optimal $\hat{\mathbf{a}}$ for this problem has been determined, the optimal apportionment is found as: $\mathbf{a} = \bar{\mathbf{a}} + \hat{\mathbf{a}}$. Since we only deal with equality constrained problems in this dissertation, constraints (15.25) and (15.26) can be written as follows:

(15.30) $$\sum_{j \in N} \hat{a}_{ij} = R_i - \sum_{j \in N} \bar{a}_{ij} \qquad \forall\ i$$

(15.31) $$\sum_{i \in M} \hat{a}_{ij} = C_j - \sum_{i \in M} \bar{a}_{ij} \qquad \forall\ j$$

[C&E] (page 429) present the following result: "Controlled roundings cannot always be obtained with prespecified roundings of the total entries". This result is of interest in chapter 16, where we use the bound vector MF-MF when solving controlled rounding problems of the type presented above. Example 15.1 presents two situations where a controlled rounding cannot be obtained with a bound vector determined by a divisor method:

Example 15.1

From chapter 3 we know that HA is the only divisor method which stays above lower quota all the time. Let us look at what may happen when a constituency bound does not stay above lower quota. We go back to Example 3.2 where a constituency with a quota of 12,05 only gets 11 seats with MF. Let us now assume that the parties in this constituency have quotas of 5,05, 4, 2, and 1 respectively. In this situation $\sum_{j \in N} \bar{a}_{ij} = 5 + 4 + 2 + 1 = 12$ and $R_i = 11$, which is a violation of constraint (15.30) because $\hat{a}_{ij} \geq 0$.

The opposite may also happen. Let us consider a situation where a constituency with a quota of 11,95 is apportioned 13 seats with MF. Furthermore, suppose that the party quotas are 4,95, 4, 2, and 1 respectively. Zero restrictedness requires that the last three parties get exactly 4, 2, and 1 seat respectively. This leaves 13 - 7 = 6 seats for the first party. However, adjacency restricts the allotment to this party to at most 5 seats. Hence, the problem has no solution, i.e. it is infeasible.

The failures in Example 15.1 are due to the predetermined bounds not staying within the quota. As we know from chapter 3, no divisor method does always stay within the quota. This brings us to the following question: Does a bound vector determined by LF or another apportionment method which stays within the quota guarantee the existence of a controlled rounding of the internal entries? The answer is no for LF, as Example 15.2 below illustrates. However, notice that an apportionment method which uses the controlled rounding formulation in section 15.1 to determine the bound vector (total entries) guarantees the existence of a controlled rounding of the internal entries.

Example 15.2

We consider the problem in Table 15.1 below which is a slightly modified version of the problem in [C&E] (page 430). The difference is that two internal entries on the diagonal have been increased by 0,01 each to get an integer value for the grand total entry.

Table 15.1

0,99	0,99	0,99	0,99	0,53	0,53	5,02
0,99	0,53	0,53	0,53	0,01	0,01	2,60
0,99	0,53	0,53	0,53	0,01	0,01	2,60
0,99	0,53	0,53	0,53	0,01	0,01	2,60
0,53	0,01	0,01	0,01	0,02	0,01	0,59
0,53	0,01	0,01	0,01	0,01	0,02	0,59
5,02	2,60	2,60	2,60	0,59	0,59	14,00

Controlled rounding problem on canonical form.

The task ahead is to round the internal entries given the bound vector LF-LF. We start by determining the bound vector. The LF apportionment to the total entries is [5, 3, 3, 3, 0, 0]. Since the matrix is symmetric, the apportionment is the same for rows and columns. In the following we focus on the rounding of the internal entries in the first row. The entries in the first four columns can at most get 1 seat each, because of the adjacency condition. Since the row bound is $R_1 = 5$, one of the internal entries in the last two columns must get 1 seat. However, this is impossible since the respective column bounds both are equal to 0.

15.4 Rounding of all or only internal entries?

Above and in earlier chapters we apportioned seats to internal entries (cells) given a predetermined bound vector. Sections 15.1 and 15.2 presented formulations where internal and total entries were apportioned seats simultaneously. In this section we compare the two ways of carrying out the apportionment. We start with an example:

Example 15.3
Consider the election situation in Table 15.2 below where three seats are to be divided between two constituencies and three parties. We are going to solve this matrix apportionment problem by both approaches mentioned above.

Table 15.2

0,81	0,35	0,35	1,51
0,49	0,50	0,50	1,49
1,30	0,85	0,85	3,00

Quota matrix for an election situation with 2 constituencies, 3 parties, and 3 seats for distribution.

We start by determining the bound vector from the total entries: All conventional apportionment methods result in the constituency bounds $\mathbf{R} = [2, 1]$, due to internal vote monotonicity, and the party bounds $\mathbf{C} = [1, 1, 1]$. The next task is to find the optimal apportionments given this bound vector. We begin with the controlled rounding (LF apportionment): There are two optimal controlled roundings of the internal entries. One of these is presented in Table 15.3. The other is found by exchanging the apportionments to the two smallest parties. As seen from Table 15.2, these two parties are identical. We are also interested in the divisor apportionments to the internal entries. It can be verified that the optimal apportionments for all constant parametric divisor methods are equal and coincide with the two optimal controlled roundings.

Table 15.3

1	1	0	2
0	0	1	1
1	1	1	3

A controlled rounding of internal entries.

Table 15.4

1	0	0	1
0	1	1	2
1	1	1	3

Controlled rounding of all entries.

The time has come to apportion seats to all entries simultaneously. It can be verified that Table 15.4 presents the optimal controlled rounding of all entries. An intuitive explanation of how this optimal controlled rounding comes about: Both small parties aspire to one seat, and based on a visual inspection of the internal entries in Table 15.2 it seems reasonable that they both win a seat in the smallest constituency. Then there is one seat left for the largest party. Clearly, this seat should be placed in the largest constituency. Notice that this solution is the same as the one obtained by giving the three largest internal entries one seat each. Moreover, the solution in Table 15.4 is also the optimal apportionment with the divisor method formulation in section 15.2 and MF as divisor method.

A conclusion to be drawn from Example 15.3 and the last part of section 15.3 is that one does not generally get the same apportionment by apportioning seats to internal and total entries simultaneously as one gets by first determining the bound vector and thereafter apportion seats to the internal entries.

Which approach is preferable, simultaneous apportionment to all entries or predetermination of a bound vector followed by apportionment to the internal entries? In Example 15.3 the outcome of the first approach, i.e. the apportionment in Table 15.4, seems to harmonize best with the quota matrix. However, notice that this apportionment gives the smallest constituency the majority of the seats, which is a violation of internal vote monotonicity for the constituency sums. The possibility of such violations of internal vote monotonicity is an argument against the simultaneous apportionment approach. However, the size of a violation is small for the controlled rounding case because the adjacency condition prevents a smaller constituency/party from getting more than one seat more than a larger one. Divisor methods do not adhere the adjacency condition, so there is another story with the divisor method formulation in section 15.2:

Example 15.4

The election situation is presented in Table 15.5. As seen from the grand total entry, there are 4 seats for distribution. These seats shall be apportioned simultaneously to all entries using the constant parametric divisor method with $t = 0{,}01$.

Table 15.5

2,99	0,01	0,01	3,01
0,33	0,33	0,33	0,99
3,32	0,34	0,34	4,00

Quota matrix for an election situation with 2 constituencies, 3 parties, and 4 seats.

Table 15.6

1	0	0	1
1	1	1	3
2	1	1	4

$CP_{0,01}$ apportionment of all entries.

By calculating the value of (15.16) for different apportionments, it can be verified that the apportionment in Table 15.6 is the optimal one. Thus, the second constituency is apportioned 2 seats more than the first constituency even though its quota is 2,02 smaller. This apportionment contrasts with the optimal cont-

rolled rounding of all entries, which gives cell (1,1) 3 seats. The use of $CP_{0,01}$ contributes to the extreme apportionment in this example; MF would have led to a more normal apportionment.

The reason why apportionments like the one in Table 15.6 may occur with the divisor method formulation in section 15.2 is that there is no condition akin to the adjacency condition built into this formulation. We see the possible occurrence of such apportionments as a serious flaw with the formulation in section 15.2. The approach with predetermination of the bound vector is therefore preferable when divisor methods are used as matrix apportionment methods. Another argument in favour of predetermination is presented below.

We rate the two controlled rounding formulations about equal. The formulation with simultaneous rounding of all entries may violate internal vote monotonicity for the total entries, while the formulation with predetermination of the bound vector may result in an infeasible problem. We recommend the approach with predetermination of the bound vector also when controlled rounding (LF) is used as matrix apportionment method. The reason for this point of view is that this approach opens for favouring of special types of constituencies and parties. As described in section 8.1 and 8.3, there are good reasons for doing so. However, be aware that such favouring usually increases the probability of the controlled rounding problem being infeasible.

15.5 Divisor methods instead of controlled rounding?

In this section we present two applications of controlled rounding and discuss briefly whether the use of a divisor method is a satisfactory way of solving these problems. Let us first recapitulate the difference between a divisor method apportionment and a controlled rounding (LF apportionment): A matrix apportionment determined by a divisor method is proportional, but may violate the adjacency condition. Conversely, a controlled rounding does satisfy the adjacency condition, but may interfere with the notion of proportionality.

An important application of controlled rounding is to control statistical disclosure in tabular presentation of frequency count data, [C&E] (page 424). Proportionality seems to be just as important as adjacency for this problem. Furthermore, the use of a divisor method instead of controlled rounding does not enhance the possibility of backtracking, i.e. determining the original counts from the resultant table. Our conclusion is that a divisor method like MF should be able to do a good job for this kind of problem.

A different kind of controlled rounding is so-called unbiased controlled rounding. In addition to the earlier requirements additivity and adjacency, a procedure for unbiased controlled rounding must satisfy the following condition: $E[\hat{a}_{ij}] = \tilde{b}_{ij}$ for all $i \in M^S$ and $j \in N^S$. [C] (page 521-522) presents an iterative procedure for unbiased controlled rounding. This procedure is probabilistic and does not usually produce the same rounding when used repeatedly for a given problem. A divisor method is deterministic in the sense that it for a given election situation always produces the same solution(s). Since the random factor is important when unbiased controlled rounding is used, divisor methods are not good tools for this kind of controlled rounding problems.

Chapter 16: Matrix bias

In this chapter we measure the bias of matrix apportionment methods. The methodology is the same as in the vector case, but with the alteration that cells are substituted for constituencies and fair shares for quotas. Section 16.1 explains why the fair shares should be used as the ideal assignment instead of the quotas. The second section presents the algorithm for determining the fair shares. This algorithm is illustrated by a numerical example in the third section. In the fourth section results of the matrix bias tests are reported. The penultimate section describes what we call the matrix bias paradox. We end the chapter with a brief discussion regarding choice of matrix apportionment method.

16.1 Ideal for the matrix apportionment

Our objective in this chapter is to measure the bias of matrix apportionment methods. To measure the vector bias in section 5.8, we compared the apportionment to the constituencies with the quotas. Recall that the house size constraint, $\sum_{i \in M} a_i = h$, was the only constraint in the free vector apportionment problem. Let us call the ratio $q_{ij} = \frac{h \cdot p_{ij}}{p_{MN}}$ **cell quota**. A way of measuring the matrix bias is to compare the apportionment to the cells with the cell quotas. However, this is not a good idea. The matrix apportionment problem has several constraints and the cell quotas along a row or column rarely sum to the integer valued bounds. If the cell quotas had been used as the standard the matrix apportionment should be compared with, the bias of the bound vector σ would have been included in the measured bias. To eliminate the impact of differences between quota totals and bounds, the ideal matrix should have the property that

its cells sum to the binding row and column bounds. Futhermore, it should be proportional to the vote matrix. The fair share matrix $\mathbf{f}^*$ has these qualities. We recapitulate from section 9.5 that when $R^0(\mathbf{p},\sigma) \neq \varnothing$, the (extended) fair share matrix is the unique solution to the allocation problem.

16.2 Determining the fair share matrix

The topic in this section is how to determine the fair share matrix for a given election situation. [B&D-2] (page 201) present an algorithm for the positive inequality constrained allocation problem. On (page 203) they conjecture that the same algorithm works for $\mathbf{p} \geq 0$ when $R^+(\mathbf{p},\sigma) \neq \varnothing$ and that it converges to the unique extended fair share matrix when $R^+(\mathbf{p},\sigma) = \varnothing$ but $R^0(\mathbf{p},\sigma) \neq \varnothing$. In this dissertation we have restricted our attention to the equality constrained problem. This simplifies the allocation (fair share) algorithm, which then is known as the RAS method. The allocation algorithm alternately scales the rows (constituencies) and columns (parties) of $\mathbf{p}$ to sum to their predetermined bounds, [Ba] (page 46). Each scaling brings us nearer the fair share matrix. The multipliers in this chapter are allocation multipliers, which are different from apportionment multipliers. For convenience we use the same notation.

The value of $\delta > 0$ can be chosen arbitrarily for the inequality constrained problem. For a given value of δ, all constituency and party multipliers are uniquely determined. However, the proof of the convergence of Balinski and Demange's algorithm places some restrictions on the choice of δ, see [B&D-2] (page 201). We now turn our attention to the equality constrained problem. The natural choice of value for δ is the national average number of seats per individual $\frac{h}{p_{MN}}$, which transforms adjusted cell votes $\delta \cdot \lambda_i \cdot p_{ij} \cdot \mu_j$ into adjusted cell quotas $\lambda_i \cdot q_{ij} \cdot \mu_j$. Even one constituency or party multiplier can be chosen arbitrarily in this case. The reason is that one equality in addition to the house size constraint is redundant for this transportation type problem. Given fixed values of δ and another multiplier, the remaining $m + n - 1$ multipliers are

uniquely determined. The fair shares will be found without fixing the value of another multiplier, so we only fix $\delta = \frac{h}{p_{MN}}$.

We now present the logic behind the allocation algorithm. Let s be a positive integer which denotes the iteration number and let $\lambda_i(s)$ and $\mu_j(s)$ denote multipliers calculated during iteration s. The adjusted quota matrix with constituency multipliers from iteration s and party multipliers from iteration a is denoted $\mathbf{q}(s,a) = (q_{ij}(s,a)) = (\lambda_i(s) \cdot q_{ij} \cdot \mu_j(a))$ where $s \in N$ and $a \in \{(s - 1), s\}$. In the explanation of the scalings we focus on constituency i. The sum of adjusted cell quotas within constituency i at the start of iteration s is:

$$(16.1) \qquad q_i(s) = \sum_{j \in N} [\lambda_i(s\text{-}1) \cdot q_{ij} \cdot \mu_j(s\text{-}1)] = \lambda_i(s\text{-}1) \cdot \sum_{j \in N} [q_{ij} \cdot \mu_j(s\text{-}1)]$$

To make the sum of adjusted cell quotas within i equal to R_i the current constituency multiplier must be scaled by the ratio $\frac{R_i}{q_i(s)}$. Thus, the formula for the updated multiplier is:

$$(16.2) \qquad \lambda_i(s) = \lambda_i(s\text{-}1) \cdot \frac{R_i}{q_i(s)} = \frac{R_i}{\sum_{j \in N} [q_{ij} \cdot \mu_j(s\text{-}1)]}$$

where the last transition follows from (16.1).

The party multipliers are determined the same way as the constituency multipliers. Once a new set of constituency or party multipliers has been determined, the adjusted quota matrix is updated. After updating for new constituency multipliers, the constituency constraints are satisfied, and after updating for new party multipliers, the party constraints are satisfied.

We need some measure to determine how close the current adjusted quota matrix is to the fair share matrix. For this purpose we calculate what we call the total **discrepancy** during iteration s, denoted $\varepsilon(s)$. This is done as follows: For each constituency we calculate how much the sum of adjusted cell quotas prior to the scaling $q_i(s)$ differs from the constituency bound R_i The resulting distance $|q_i(s) - R_i|$ is called the discrepancy for constituency i. We calculate the

discrepancy for a party similarly. The total discrepancy $\varepsilon(s)$ is the sum of all constituency and party discrepancies during iteration s. Once $\varepsilon(s)$ becomes smaller than or equal to the predecided tolerance level $\bar{\varepsilon}$, we declare that an adjusted quota matrix close enough to the fair share matrix has been found. In our Pascal program of the allocation algorithm, which is a part of the MatrixBias program presented in Appendix 2, we apply $\bar{\varepsilon} = 0{,}0001 = 1 \cdot 10^{-4}$.

All parts of the allocation algorithm for the equality constrained problem have now been introduced, so it is time to present the algorithm itself:

Algorithm 16.1

Step 1: This is the initialization step. The initial multipliers are $\lambda_i(0) = \mu_j(0) = 1$ for all i and j. Furthermore, we set $s = 1$.

Step 2: The total discrepancy is equal to zero at the start of each iteration, so we set $\varepsilon(s) = 0$. For each constituency we find the sum of adjusted cell quotas within the constituency:

$$q_i(s) = \lambda_i(s\text{-}1) \cdot \sum_{j \in N} [q_{ij} \cdot \mu_j(s\text{-}1)] \tag{16.3}$$

and calculate the new constituency multiplier:

$$\lambda_i(s) = \lambda_i(s\text{-}1) \cdot \frac{R_i}{q_i(s)} \tag{16.4}$$

The final part of the step is the calculation of the total discrepancy for the constituencies: $\varepsilon(s_M) = \sum_{i \in M} | q_i(s) - R_i |$.

Step 3: For each party we find the sum of adjusted cell quotas within the party:

$$q_j(s) = \mu_j(s\text{-}1) \cdot \sum_{i \in M} [q_{ij} \cdot \lambda_i(s)] \tag{16.5}$$

and calculate the new party multiplier:

$$(16.6) \qquad \mu_j(s) = \mu_j(s\text{-}1) \cdot \frac{C_j}{q_j(s)}$$

Then we calculate the total discrepancy for the parties: $\varepsilon(s_N) = \sum_{j \in N} | q_j(s) - C_j |$.

Step 4: We use the multipliers calculated in steps 2 and 3 to update the adjusted quota matrix. The updated matrix is found as $\mathbf{q}(s,s) = (\lambda_i(s) \cdot q_{ij} \cdot \mu_j(s))$. We find the total discrepancy during iteration s by adding the discrepancies from steps 2 and 3: $\varepsilon(s) = \varepsilon(s_M) + \varepsilon(s_N)$. The calculated $\varepsilon(s)$ is compared with the predecided tolerance level $\bar{\varepsilon}$. If $\varepsilon(s) \leq \bar{\varepsilon}$, we stop because a matrix close enough to the fair share matrix has been found, otherwise we increase s by one and return to step 2.

The allocation algorithm is illustrated by a numerical example in the next section. Now to a result regarding the algorithm: It converges to the unique (extended) fair share matrix for non-negative equality constrained problems if and only if (16.7) holds, see [Ba] (page 51→).

$$(16.7) \qquad p_{I\underline{J}} = 0 \quad \text{implies} \quad R_I \leq C_J \quad \text{and} \quad R_{\underline{I}} \geq C_{\underline{J}}$$

(16.7) is nothing else than the first two supply-demand conditions from (9.11). The third condition in (9.11) is redundant here because we face an equality constrained problem. Recall that the conditions in (9.11) were necessary and sufficient for $R^0(\mathbf{p},\sigma)$ to be non-empty. As Example 9.2 and Example 9.3 demonstrated, the conditions in (16.7) might be violated. However, we have not encountered any election situation where $R^0(\mathbf{p},\sigma) = \varnothing$. We base this statement on the fact that the apportionment algorithm from chapter 12 has been able to solve all 312 election situations (104 data sets, each with 3 bound vectors) it has been confronted with.

Except for the stop criterion, Algorithm 16.1 is equal to the algorithm in [B&S] (page 351). Their stop criterion demands that the change in the party multiplier $|\mu_j(s) - \mu_j(s\text{-}1)|$ shall be sufficiently small for all j. While their primary concern

is to determine the multipliers, our main task is determination of the fair shares, which explains our stop criterion. [B&S] (page 350-351) deal with the positive equality constrained problem. They explain that each execution of step 2 and 3 will bring us nearer the fair share matrix. This means that the total discrepancy will decrease from one iteration to the next, i.e. $\varepsilon(s+1) < \varepsilon(s)$, for positive equality constrained problems. We conjecture that the same holds for equality constrained problems where $\mathbf{p} \geq 0$ and $R^0(\mathbf{p},\sigma) \neq \varnothing$.

The discrepancies calculated during steps 2 and 3 of the allocation algorithm show the discrepancy prior to the adjustments in these steps. Because the adjustments bring us closer to the fair share matrix, the real total discrepancy is smaller than the calculated $\varepsilon(s)$ we compare with $\bar{\varepsilon}$. Notice also that each cell is counted twice in $\varepsilon(s)$, but not with the same discrepancy. Let us define cell discrepancy as the deviation between the content of a cell and the fair share for this cell. Based on the explanation above, we conclude that the maximal cell discrepancy in the final adjusted quota matrix is well below $\bar{\varepsilon}$.

We end this section with some data regarding the performance of the allocation algorithm: For the 23 election situations utilzed in the main matrix bias test, the algorithm used an average of 6,78 iterations to find an adjusted quota matrix which satisfied the tolerance level $\bar{\varepsilon} = 1 \cdot 10^{-4}$. The iteration numbers for these election situations are presented in Appendix 4. With one exception, the number of iterations ranged from 4 to 11 with a median of 6. The exception is the election situation from Norway in 1969, which took 16 iterations to solve. We have not investigated why this election situation of moderate size was much more troublesome than the rest.

16.3 Fair share example

In this section we illustrate how the allocation algorithm, i.e. Algorithm 16.1, determines the fair shares. The basis for the example is the election situation in Table 16.1. This is the same election situation we made use of in chapter 13.

Table 16.1

i|*j*	B	D	G	Votes	Seats
Rev	9743	27736	9440	46919	12
NoE	6015	4606	2741	13362	5
Aul	3668	1760	1257	6685	5
Votes	19426	34102	13438	66966	
Seats	7	11	4		22

Vote matrix for a 3 × 3 allocation problem.

Before Algorithm 16.1 can be applied, the vote matrix must be converted to a quota matrix. This conversion is carried out by multiplying each vote figure in Table 16.1 by $\frac{h}{p_{MN}} = \frac{22}{66966} \approx 3{,}28525 \cdot 10^{-4}$. Table 16.2 presents the resulting quota matrix with three decimals accuracy. NB. In all calculations in this section we operate with a finer accuracy than the three decimals shown. Some places, most notably in (16.10), this leads to some seemingly peculiar roundings. The tolerance level remains to be decided. We choose $\bar{\varepsilon} = 0{,}01 = 1 \cdot 10^{-2}$, i.e. a higher tolerance level than the one applied in the Pascal program.

Table 16.2

i|*j*	B	D	G	Sum	Seats
Rev	3,201	9,112	3,101	15,414	12
NoE	1,976	1,513	0,900	4,390	5
Aul	1,205	0,578	0,413	2,196	5
Sum	6,382	11,203	4,415	22,000	
Seats	7	11	4		22

Quota matrix for the allocation problem in Table 16.1.

A quota matrix can be expressed as $\mathbf{q} = \delta \cdot \mathbf{p}$, where $\delta = \frac{h}{p_{MN}}$. As explained in the previous section, we fix δ at $\frac{h}{p_{MN}}$. We start Algorithm 16.1 by initializing the multipliers, $\lambda_i(0) = \mu_j(0) = 1$ for all i and j, and setting the iteration number s to 1.

We move on to step 2 of the algorithm. The total discrepancy during the first iteration is initialized by $\varepsilon(1) = 0$. Our next task is the calculation of sums of adjusted cell quotas within the three constituencies. Since all constituency and party multipliers are equal to 1 at this stage, the $q_i(1)$s are simply the numbers in the sum column in Table 16.2. To illustrate how the sums of adjusted cell quotas are calculated, we find the sum within constituency Rev:

$$(16.8) \qquad q_{Rev}(1) = \lambda_{Rev}(0) \cdot [q_{Rev.B} \cdot \mu_B(0) + q_{Rev.D} \cdot \mu_D(0) + q_{Rev.G} \cdot \mu_G(0)] \approx 1 \cdot (3{,}201 \cdot 1 + 9{,}112 \cdot 1 + 3{,}101 \cdot 1) = 15{,}414$$

The discrepancy for Rev is the absolute deviation between $q_{Rev}(1)$ and the constituency bound R_{Rev}: $|15{,}414 - 12| = 3{,}414$. Moreover, the total discrepancy for the constituencies is $\varepsilon(1_M) = 3{,}414 + |4{,}390 - 5| + |2{,}196 - 5| = 6{,}828$. What remains to be done in step 2 is the determination of new constituency multipliers. Formula (16.4) gives the following new value for Rev's multiplier:

$$(16.9) \qquad \lambda_{Rev}(1) = \lambda_{Rev}(0) \cdot \frac{R_{Rev}}{q_{Rev}(1)} \approx 1 \cdot \frac{12}{15{,}414} \approx 0{,}779$$

The new values of λ_{NoE} and λ_{Aul} are found similarly. They are $\lambda_{NoE}(1) \approx 1{,}139$ and $\lambda_{Aul}(1) \approx 2{,}277$ respectively. With these new constituency multipliers, the adjusted quota matrix becomes as presented in Table 16.3:

Table 16.3

	μ	1,000	1,000	1,000			
λ	*i\j*	B	D	G	Sum	Bound	Discrepancy
0,779	Rev	2,492	7,094	2,414	12,000	12	0,000
1,139	NoE	2,251	1,724	1,026	5,000	5	0,000
2,277	Aul	2,743	1,316	0,940	5,000	5	0,000
	Sum	7,486	10,134	4,380	22,000		0,000
	Bound	7	11	4		22	
	Discrep.	0,486	0,866	0,380	1,733		
New	μ	0,935	1,085	0,913			

Adjusted quota matrix given δ, $\lambda_i(1)$, and $\mu_j(0)$. The new party multipliers $\mu_j(1)$ are shown below the main body of the table.

From the sum column in Table 16.3 we observe that the sums of adjusted cell quotas within the constituencies now are equal to the bounds. However, the sums within the parties are not, and that is what we are going to deal with in step 3 of the algorithm. Let us first demonstrate how the sum within party B has been calculated:

(16.10)
$$\begin{aligned} q_B(1) = \mu_B(0) \cdot [q_{Rev.B} \cdot \lambda_{Rev}(1) + q_{NoE.B} \cdot \lambda_{NoE}(1) + q_{Aul.B} \cdot \lambda_{Aul}(1)] \approx \\ 1 \cdot (3{,}201 \cdot 0{,}779 + 1{,}976 \cdot 1{,}139 + 1{,}205 \cdot 2{,}277) \approx \\ 2{,}492 + 2{,}251 + 2{,}743 = 7{,}486 \end{aligned}$$

This sum results in a discrepancy of $|7{,}486 - 7| = 0{,}486$ for B. The total discrepancy for the parties is $\varepsilon(1_N) = 1{,}733$, as shown in Table 16.3. With the help of formula (16.6), we find the value of B's multiplier which eliminates the discrepancy for B:

(16.11)
$$\mu_B(1) = \mu_B(0) \cdot \frac{C_B}{q_{B(1)}} \approx 1 \cdot \frac{7}{7{,}486} \approx 0{,}935$$

The new multipliers for D and G are found similarly. They are $\mu_D(1) \approx 1{,}085$ and $\mu_G(1) \approx 0{,}913$ respectively, as shown at the bottom of Table 16.3. We use these new party multipliers to update the adjusted quota matrix. The matrix after the first iteration can be expressed as $\mathbf{q}(1,1) = (\lambda_i(1) \cdot q_{ij} \cdot \mu_j(1))$.

Step 4 of the algorithm determines whether the current adjusted quota matrix is close enough to the fair share matrix. The total discrepancy during the first iteration is $\varepsilon(1) = \varepsilon(1_M) + \varepsilon(1_N) \approx 6{,}828 + 1{,}733 = 8{,}561$. This is far above the predecided tolerance level of $\bar{\varepsilon} = 0{,}01$, so another iteration must be carried out. The iteration number s is therefore increased by one to 2 and steps 2 through 4 of Algorithm 16.1 are repeated.

We do not show the calculations in the second or any of the following iterations here. The demonstration above should have made it clear how these calculations are carried out. Table 16.4 shows the multipliers and discrepancies along the way:

Table 16.4

Iteration	λ			μ			ε		
	Rev	NoE	Aul	B	D	G	*M*	*N*	Total
0	1,000	1,000	1,000	1,000	1,000	1,000			
1	0,779	1,139	2,277	0,935	1,085	0,913	6,828	1,733	8,561
2	0,764	1,159	2,346	0,926	1,093	0,913	0,470	0,142	0,612
3	0,762	1,162	2,354	0,925	1,093	0,913	0,052	0,016	0,068
4	0,762	1,162	2,355	0,925	1,093	0,913	0,006	0,002	0,008

Constituency multipliers, party multipliers, and discrepancies in the different iterations.

As seen from the table, the predecided tolerance level for maximal error during an iteration is satisfied after the fourth iteration. The final adjusted quota matrix **q**(4,4), which we might term the fair share matrix, is presented in Table 16.5:

Table 16.5

	μ	0,925	1,093	0,913			
λ	*i**j*	B	D	G	Sum	Bound	Discrepancy
0,762	Rev	2,254	7,589	2,157	12,000	12	0,000
1,162	NoE	2,123	1,922	0,955	5,000	5	0,000
2,355	Aul	2,623	1,489	0,888	5,000	5	0,000
	Sum	7,000	11,000	4,000	22,000		0,001
	Bound	7	11	4		22	
	Discrep.	0,000	0,000	0,000	0,000		

Fair share matrix and multipliers for the allocation problem in Table 16.1.

Let us compare the fair share matrix with the HA apportionment found in chapter 13. The largest deviation between the HA apportionment, shown in Table 16.6 below, and the fair share matrix is 0,489 = | 1,489 - 1 | and occurs for the cell Aul.D.

Table 16.6

*i**j*	B	D	G
Rev	2	8	2
NoE	2	2	1
Aul	3	1	1

HA apportionment for the problem in Table 16.1.

The example above gives us an idea: Even if there is no direct connection between allocation and apportionment multipliers, an interesting possibility is to use the multipliers from the allocation problem as initial multipliers for the apportionment algorithm in chapter 12. We have not tested this idea.

16.4 Matrix bias tests

In our test of matrix bias we build on the methodology introduced in chapter 5 with the change regarding the ideal assignment described in section 16.1. Another difference compared with the vector bias test in section 5.8 is that we now measure bias between groups of cells instead of between groups of constituencies. This means an increase in the number of items from m to $m \cdot n$. However, the maximal number of items among the utilized data sets does not increase in that proportion because we have not got matrix data for the countries with the largest number of constituencies in the vector test. The maximal number of cells among the matrix data sets is $36 \cdot 13 = 468$ for the German set from 1919. However, this election situation contains many zero cells, so the number of non-zero cells is lower. For comparison, the Japanese data set with its 130 constituencies had the maximal number of items in the vector bias test.

We carry out the main matrix bias test with data from the Nordic countries plus Germany, Austria, and Luxembourg. To reduce the workload, we utilize only 23 out of the 104 data sets available. The principal criterion when picking these 23 sets was that they should be spread out in time. This in an attempt to avoid auto correlation between utilized data sets. Another consideration in the selection process was that the number of data sets from a country should stand in proportion to the number of available sets from this country. Let us explain how we went about picking data sets: For each country we started by picking the latest of the available data sets. The only exception to this rule is Luxembourg where the penultimate set was chosen. Of the countries in the test, Luxembourg and Austria are the ones with fewest available data sets. We utilize only one set from

each. For each of the other countries we picked out the data set subsequent to World War 2 plus a set about midway between the two sets picked so far. Furthermore, we utilize the German set from 1919. Our collection of Swedish data sets goes back to 1911. We have picked out the 1911 set and a set about midway between the 1911 and the after World War 2 set. The 23 data sets thus chosen should be a good foundation for measuring the "universal" matrix bias. More information about the chosen data sets can be found in Appendix 4. In addition to the main bias test, we carry out a test based on all Icelandic data sets. The purpose of this test is to study the matrix bias within a country.

We apply the bound vector MF-MF for every data set in the tests. This means that we do not test whether the bound vector applied has any impact on the matrix bias. However, there is no reason for believing that bound vectors have a great impact on the matrix bias for divisor methods since, by definition, the fair share matrix is proportional to the vote matrix given any bound vector. We include five apportionment methods in the matrix bias tests; the four divisor methods $CP_{0,01}$, DM, MF, and HA plus LF. What we call LF is controlled rounding of the internal entries. The LF apportionment (controlled rounding) for each election situation is found by solving a LP problem of the type presented in section 15.3. As mentioned in that section, there may not exist a LF apportionment given predetermined bounds. We did not encounter any such instance among the 31 election situations utilized. However, the bound vector MF-MF does not put the LF apportionment to the test. For further comments on this matter, we refer to section 16.6.

We divide the cells in 2, 3, and 4 groups using number, size, cluster, and quota division. The complete results of the matrix bias tests are presented in Appendix 4. Table 16.7 below captures the interesting points of the main matrix bias test. It shows the estimated bias percentages with division in 3 groups. We have excluded the bias percentages with number division from the table because this division method is even worse here than it was in the vector case. The shortcoming of number division is that it classifies far too few cells as small. A way of making it work better is to eliminate all zero cells before the division, but we do not believe

that this is enough to get a satisfactory division.

The matrix bias figures we got while testing the Pascal program were ambiguous regarding the direction of the bias for the different apportionment methods. We therefore decided to employ two-sided t-tests for all 5 apportionment methods, i.e. let the alternative hypothesis be $\varepsilon_{DE} \neq 0$.

Table 16.7

Division	$\bar{\varepsilon}$	LF	$CP_{0,01}$	DM	MF	HA
Size	(L,M)	-0,8%	9,1% **	0,9%	-4,3% *	-8,0% **
	(L,S)	-0,8%	0,5%	-3,4% *	-5,2% **	-4,6%
	(M,S)	0,0%	-10,8% **	-4,4% **	-1,0%	2,8%
Cluster	(L,M)	-0,8%	9,0% **	0,4%	-4,5% **	-8,8% **
	(L,S)	-1,7%	-10,8% *	-5,4% **	-4,5% **	1,9%
	(M,S)	-0,9%	-22,8% **	-6,1% **	-0,1%	9,7% **
Quota	(L,M)	-0,5%	6,1% **	-1,4%	-2,8% *	-7,1% **
	(L,S)	-0,7%	-9,2% **	-3,5% **	-2,0% *	2,5%
	(M,S)	-0,2%	-17,2% **	-2,2%	0,6%	8,8% **

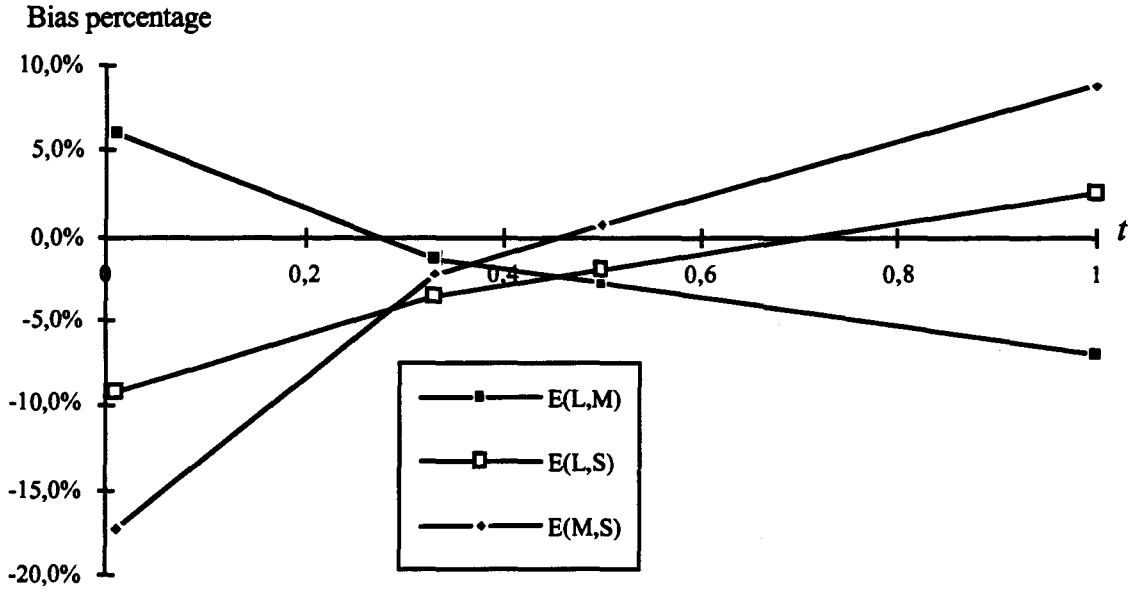
Main matrix bias test:
Estimated bias percentages between the groups of large (L), medium (M), and small (S) cells.
** = Significant at the 1%-level * = Significant at the 5%-level (with two-sided t-test)

Compared with the vector bias results, some of the estimates in Table 16.7 are surprising. One thing which has not changed is that LF seems to be unbiased. Although all LF percentages are negative, these estimates are far from significant. The other unbiased vector method, MF, has difficulties with maintaining its status here. It seems to treat the groups of medium and small cells about equally. However, the other bias percentages, of which some are significant at the 1%-level, tell us that MF disfavours the group of large cells. The greatest surprise in Table 16.7 is the bias percentages between the groups of large and medium sized cells for $CP_{0,01}$ and HA. $CP_{0,01}$ favours large cells while HA disfavours them, with all estimates significant at the 1%-level. This result is just the opposite of what we experienced in the vector bias test. We call it the **matrix bias paradox**. An explanation of why the paradox occurs follows in the next section. Regarding $\varepsilon_{(L,S)}$ and $\varepsilon_{(M,S)}$, things are more normal for $CP_{0,01}$. With one exception, $CP_{0,01}$ favours the group of small cells. The only thing which is normal regarding HA is the estimates for $\varepsilon_{(M,S)}$ Two of these estimates are

significant at the 1%-level, and they tell us that HA favours the group of medium sized cells in the comparison with the group of small cells. The last apportionment method in the test, DM, produces results which are a kind of mirror image of MF's. While MF seems to be approximately unbiased in the comparison of medium and small sized cells, DM seems to have this property in the comparison between large and medium sized cells. A majority of DM's bias percentages involving the group of small cells are significant. DM disfavours this group of cells.

Four of the apportionment methods in Table 16.7 are constant parametric divisor methods. They can be written as $CP_{0,01}$, $CP_{\frac{1}{3}}$, $CP_{0,5}$, and CP_1 respectively. To study the relationship between the parameter t and the bias percentages, we illustrate the bias percentages with quota divison graphically:

Figure 16.1



Bias percentages as a function of t with quota division in three groups.

As can be read out of Table 16.7, the graphs with cluster and size division would not have been that different from Figure 16.1. Below we describe the functional relationship between t and the bias percentages in Table 16.7. We start with $\varepsilon_{(L,M)}$. With all three division methods $\varepsilon_{(L,M)}$ is a decreasing function of t. It is positive for $t = 0,01$ and changes from positive to negative around $t = \frac{1}{3}$ some-

where. With the exception of $\varepsilon_{(L,S)}$ with size division, $\varepsilon_{(M,S)}$ and $\varepsilon_{(L,S)}$ are increasing functions of t. The slope of $\varepsilon_{(M,S)}$ is steeper than the slope of $\varepsilon_{(L,S)}$. Both $\varepsilon_{(M,S)}$ and $\varepsilon_{(L,S)}$ are negative for $t = 0{,}01$. $\varepsilon_{(M,S)}$ becomes positive when t is about 0,5, while $\varepsilon_{(L,S)}$ does the same some place between $t = 0{,}5$ and $t = 1$.

Table 16.8

Division	$\overline{\varepsilon}$	LF	$CP_{0,01}$	DM	MF	HA
Size	(L,M)	0,1%	12,9% *	4,7%	-5,3%	-4,5%
	(L,S)	-1,3%	5,5% *	-1,7%	-8,1% *	-7,2% *
	(M,S)	-1,6%	-12,3%	-7,4% *	-3,3%	-3,0%
Cluster	(L,M)	-1,6%	13,1% **	2,3%	-6,6%	-5,8% *
	(L,S)	-1,9%	-3,8%	-2,6%	-7,5% *	-4,3% *
	(M,S)	-0,4%	-20,8% **	-5,3% **	-1,4%	1,1%
Quota	(L,M)	-1,3%	9,3% *	-1,5%	-4,8%	-6,8% *
	(L,S)	-0,9%	-5,5% *	-3,6% *	-3,3% *	-3,0%
	(M,S)	0,2%	-17,7% **	-2,4%	0,9%	3,1%

Icelandic matrix bias test:
Estimated bias percentages between the groups of large (L), medium (M), and small (S) cells.
** = Significant at the 1%-level * = Significant at the 5%-level (with two-sided t-test)

Table 16.8 presents the results of the Icelandic matrix bias test. This test is based on 11 data sets. Compared with the main matrix bias test, fewer estimates are significant. This may be due to the lower number of observations here. LF still seems to be unbiased. As in the main test, most of the bias percentages for LF are negative. Every bias percentage for $CP_{0,01}$ and DM has the same sign as in the main test. However, many estimates are less significant here. MF still seems to disfavour the group of large cells, but only some estimates are significant. We do not present graphs of the bias percentages here. However, notice that $\varepsilon_{(L,M)}$ and $\varepsilon_{(L,S)}$ with both size and cluster division have their bottom point for $t = 0{,}5$. Finally, even though the estimates for $\varepsilon_{(L,M)}$ with HA and $CP_{0,01}$ are less significant than in the main test, the matrix bias paradox is still present. Below we reveal why the paradox is not so strange after all:

16.5 Explanation of the matrix bias paradox

What we call the matrix bias paradox is that $CP_{0,01}$ favours large cells while HA disfavours them in the comparison with medium sized cells. This is in complete contrast to the vector bias results in section 5.8. The constituency and party bounds are factors which are not present in the free vector apportionment problem. They play an important role in the explanation of the matrix bias paradox.

Let m_j denote the number of constituencies party j participates in. We split the parties in two categories based on the magnitude of the party bounds. Parties for which $C_j \leq m_j$ are called small while the other parties are called large. We also find it useful to distinguish between constituencies of different sizes. Based on an imaginary cluster/quota division, we place the constituencies in three groups. We use the classifications of parties and constituencies to present a rough classification of the members of the three cell groups in Table 16.7: The group of large cells (L) consists of cells from large parties in large constituencies. Members of the group of medium sized cells (M) are cells from small parties in large constituencies and from large parties in medium sized constituencies. The remaining combinations of party and constituency sizes make up the group of small cells (S). Hence, S consists of cells from large parties in small constituencies and from small parties in medium and small constituencies. In the following we utilize this classification of the cells plus our vector bias knowledge.

We start by analysing what happens with $CP_{0,01}$. The first quotient in a cell $\ln(\frac{p_{ij}}{0,01})$ is much larger than the second one $\ln(\frac{p_{ij}}{1,01})$ with this divisor method. This size difference has a strong influence on the apportionments for the small parties. A small party shall at most have m seats. Its second quotients, even in large constituencies, seldom win a seat. Thus, a small party usually gets one seat in every constituency it becomes represented. Large parties get more than one seat in at least one constituency. The impact of the size difference between first and second quotients becomes less important the more seats C_j such a party shall

have. Despite facing a matrix problem, which means simultaneous apportionment, we base our explanation on the simplifying notion that seats are apportioned within each party and that small parties are apportioned first. This should be a reasonable simplification according to the arguments presented above. In the following we assume that a small party has about the same percentage support in every constituency it participates. Usually, this is not far away from reality. Exceptions are parties with regional strongholds. The $CP_{0,01}$ apportionment within a small party can be characterized as "one seat to as many constituencies as possible". Compared with the fair shares, this apportionment often distributes too many seats to cells representing medium sized constituencies. This leaves too few seats to cells from large constituencies. Recall that cells from large constituencies within small parties were classified as members of the group of medium sized cells (M). In many Nordic data sets about 50% of the parties are small, so the accumulated disfavouring of cells from M within these parties is important for the total matrix bias. Be aware that the description above does not suit very small parties because such parties usually win all their seats in large constituencies. However, this is not a problem because all cells from very small parties normally belong to the group of small cells (S).

The constituency bounds fix how many seats each constituency gets. When the small parties get too few seats in the large constituencies compared with their fair shares there, the large parties must fill the fair share gap left behind. Conversely, when the small parties are apportioned too many seats in the medium sized constituencies, there are too few seats left for the large parties. The result within the large parties is that $CP_{0,01}$ usually apportions more seats than the fair shares suggest to cells from large constituencies. This favouring is at the expense of cells from medium sized constituencies. Together with the disfavouring of cells belonging to group M within the small parties, this explains why $CP_{0,01}$ favours L compared with M.

Next we analyse what happens with HA. In this case it is not obvious that we can assume that seats are apportioned within each party and that small parties are apportioned first. Since large size differences between marginal quotients still

appear most frequently within small parties, we find it reasonable to continue to do so. Because HA favours large entities in the vector case, the small parties often get more than their fair share of seats in the large constituencies. This leaves too few seats to the large parties in these constituencies. Conversely, the small parties are usually apportioned too few seats among the medium sized constituencies, which results in the large parties getting too many seats in these constituencies. Thus, the main beneficiaries within both small and large parties are cells belonging to M, with the end result that HA favours medium sized cells (M) compared with large cells (L).

We end this section with a brief reflection regarding matrix bias. In addition to the matrix apportionment method applied, there are two factors which influence the matrix bias, namely the election situations utilized and the division method applied. To see that the types of election situations utilized have an effect, just compare the bias percentages in the Icelandic test with those in the main bias test. Division methods have not been discussed much in this chapter, but their influence is important. For instance, the bias percentages with number division were quite different from the ones presented in section 16.4.

16.6 Choice of matrix apportionment method

Which matrix apportionment method is preferable? Matrix bias is an important factor in this evaluation. We regard unbiasedness in all pairwise group comparisions as the ideal. Based solely on the results of the matrix bias tests, LF (controlled rounding) is the best matrix apportionment method. However, the use of the bound vector MF-MF in these tests is to the advantage of LF. The explanation is as follows: LF (controlled rounding) in the matrix bias tests was based on the quota matrix. Because MF-MF is a bound vector which is close to the quota totals, the cell quotas were close to the fair shares in the utilized election situations. This resulted in feasible controlled rounding problems for all 31 election situations encountered and low bias percentages for LF. We next

explain the consequences of bound vectors which deviate largely from the quota totals: Many cell quotas will deviate significantly from the fair shares with such bound vectors. Since the adjacency condition is connected to the cell quotas when LF (controlled rounding) is based on the quota matrix, the result will often be infeasible controlled rounding problems. Moreover, when the problems are feasible, the bias percentages will usually be higher than with MF-MF.

The drawback of basing controlled rounding on the quota matrix is that this approach may result in infeasible problems for the internal entries, see section 15.3. An alternative way of carrying out controlled rounding (LF) for matrix apportionment problems is to base the rounding (apportionment) on the fair share matrix instead. In hindsight, this would have been a better approach because of the following result:

Corollary 16.1
For a fair share matrix which bounds (total entries) all are integer valued, there always exists a controlled rounding of the internal entries.

This corollary follows from Theorem 15.1. When all bounds (total entries) for a fair share matrix are integer valued, adjacency fixes the values of these entries. The actual bound vector is therefore the only possible "rounding" of the total entries. Theorem 15.1 states that there always exists a controlled rounding of all entries. Such a controlled rounding must "round" the total entries the way described above, which further means that there must exist a controlled rounding of the internal entries given this bound vector.

Controlled rounding (LF) based on the fair share matrix seems like a reasonable apportionment method for the matrix apportionment problem. However, like LF in the vector case, controlled rounding violates the important proportionality condition/axiom of consistency. If the principle of proportionality is seen as too important to ignore, we are left with divisor methods satisfying the six matrix apportionment axioms in chapter 9. The extremities in our tests, $CP_{0,01}$ and HA, are not good choices. HA is not attractive because too many of the small parties'

seats are placed in large constituencies, while the drawback of $CP_{0,01}$ is that it spreads their seats too much. In other words, HA gives a too narrow and $CP_{0,01}$ a too broad constituency basis for a small party in relation to its support. A divisor method somewhere between these two is therefore desirable. The two candidates we have investigated are DM and MF. As mentioned in section 16.4, DM seems to favour the group of small cells at the expense of the two other cell groups, while MF seems to disfavour the group of large cells to the gain of the two other cell groups. Since unbiasedness seems to be unachievable, we introduce a second criterion for ranking the divisor methods: We find it desirable that each constituency is represented by many different political opinions. In other words, to be acceptable a divisor method should not disfavour small cells. Both DM and MF satisfy this criterion. The choice of a divisor method is not restricted to DM, MF, or a constant parametric divisor method for that matter. In fact, the illustration of bias percentages as functions of t in Figure 16.1 indicates that it is difficult, if not impossible, to find a method within the constant parametric family which is approximately unbiased in all pairwise group comparisons. A more thorough investigation of matrix bias with different constant parametric divisor methods should give more insight into the behaviour of the bias percentages as functions of t. This matter is of special interest within a single country. Maybe it is possible to construct a divisor method which is approximately unbiased over a collection of relevant data sets?

Chapter 17: Decomposition

In this chapter we decompose a set of suitable constituency multipliers into 3 components. The idea behind the decomposition is to throw light on different aspects of the actual matrix apportionment. We assume throughout the chapter that we face a matrix apportionment problem with equality party constraints and where all party multipliers μ_j are set equal to 1. The structure of the chapter is as follows: Section 17.1 presents the 4 multiplier sets utilized, while the second section explains how the 3 components are derived from these sets. In section 17.3 we describe the consequences of normalizing the multiplier sets. The decomposition process is illustrated by an example in the final section.

17.1 Utilized multiplier sets

We utilize 4 multiplier sets in the decomposition process; the three sets of initial constituency multipliers from section 11.3 together with a suitable multiplier set. There are several suitable sets to choose from. We have chosen the suitable set found when the apportionment initialization multiplier set is the starting point and ρ-effect selection is used during the solution process. The background for this choice is as follows: Measured by both initial measure of goodness and number of iterations, the apportionment initialization set is the initial multiplier set which on average is closest to a suitable multiplier set. Based on the same criteria, the quota ratio initialization set is closer to a suitable set than the no initialization set. Thus, the no initialization set is usually farthest from a suitable set. This explains why we let the ordering of the multiplier sets be as presented in (17.1). However, since this ordering is based on the normal relationship between the sets, there are

election situations where another ordering would have been more appropriate.

(17.1) No - Quota ratio - Apportionment - Suitable

17.2 The components

We define a component as the ratio between two successive multiplier sets in (17.1), with the set to the right in the numerator. Thus, there are three components. They are described below. Each component is a vector of length m, even if we below focus on components for a single constituency. We give the components names based on our interpretation of what kind of adjustment they represent.

The first component is derived from the first two sets in (17.1). A brief recapitulation of the characteristics of these multiplier sets: All constituency multipliers are equal to 1 with no initialization, while quota ratio initialization makes sure that every constituency has the same adjusted average number of people per seat.

(17.2) $$\text{Bound component} = \frac{\lambda_i\,(\text{Quota ratio})}{\lambda_i\,(\text{No})} = \frac{\lambda_i\,(\text{Quota ratio})}{1} = \lambda_i\,(\text{Quota ratio})$$

We use the name **bound component** because the quota ratio multiplier set levels out differences between constituencies caused by the determination of constituency bounds. Thus, the bound component tells something about the bias inherent in the constituency bounds.

The target quotients, which are a sort of marginal quotients for the constituencies, play an important role in apportionment initialization. Apportionment initialization adjusts the constituency multipliers such that all target quotients become equal. Since this is an attempt to even out differences between election situations within different constituencies, we call the component derived from the

second multiplier ratio **constituency component**:

(17.3) $$\text{Constituency component} = \frac{\lambda_i \text{ (Apportionment)}}{\lambda_i \text{ (Quota ratio)}}$$

In connection with this and the next component it is important to be aware of the following fact: The magnitude of a constituency multiplier determined by apportionment initialization depends on the divisor method via the target quotient t_i. Moreover, it also depends on the value chosen for the initial marginal value of representation $\mathring{\kappa}$. Let us review how different constant parametric divisor methods influence the target quotients: A low parameter value t results in larger quotients $\ln(\frac{p_{ij}}{l-1+t})$ and thereby larger target quotients than a high t value. We see from equation (11.26) that a target quotient larger than $\mathring{\kappa}$ results in a multiplier value smaller than 1. Let us take a closer look at what happens when a very low parameter value like $t = 0{,}01$ is used: The result is a wide gap between quotients because the first quotients $\ln(\frac{p_{ij}}{0{,}01})$ are very much larger than the other quotients. Moreover, when quotients within a constituency are ranked, the usual situation is that every first quotient is larger than the largest party's second quotient. These relationships have the following consequences for the apportionment initialization: The target quotient is a weighted average of two quotients. For constituencies where the number of parties is larger than the number of seats, i.e. $n > R_i$ both these quotients are very large. Furthermore, for constituencies where $n = R_i$, one quotient is very large and the other is of normal size. Finally, when $n < R_i$, both quotients are of normal size. In these three cases the target quotient becomes very big, big, and normal respectively. This leads to very small multipliers when $R_i < n$, small multipliers when $R_i = n$, and normal multipliers when $R_i > n$. The end result is often an apportionment initialization set with large differences in the relative sizes of the multipliers.

The third component stems from the integration of all constituency election situations into the matrix apportionment. This is the reason for the name **matrix component**:

(17.4) $$\text{Matrix component} = \frac{\lambda_i \text{ (Suitable)}}{\lambda_i \text{ (Apportionment)}}$$

The matrix component for a constituency which multiplier has not been adjusted during the solution process is equal to 1. The matrix component is unique because we in section 17.1 prescribed how the suitable multiplier set shall be determined from the set of apportionment initialization multipliers. To shorten the component names, we omit the word "component" some places in the following. Then we are ready to present the decomposition of the suitable multiplier set:

(17.5) λ_i (Suitable) = Bound · Constituency · Matrix

On the right hand side of (17.5) we have utilized that a no initialization multiplier λ_i (No) always is equal to 1.

17.3 Normalization of multiplier sets

A set of constituency multipliers can be scaled by the positive factor ζ to attain any wanted magnitude. The new multiplier values are $\zeta \cdot \lambda_i$ for all i. Since all multipliers are scaled by the same factor, the new multiplier set $\zeta \cdot \lambda$ implies the same assignment as before the scaling. A multiplier set is normalized by scaling it such that the geometric or arithmetic mean of all multipliers become equal to 1. The multiplicative normalization factor is found as the inverse of the geometric mean of the multipliers:

(17.6) $$\frac{1}{\theta} = \left(\prod_{i \in M} \lambda_i\right)^{\frac{1}{m}}$$

while what we call the additive normalization factor is found as the inverse of the arithmetic mean:

(17.7) $$\frac{1}{\theta} = \frac{1}{m} \cdot \sum_{i \in M} \lambda_i$$

As seen from (17.6) and (17.7), we use the notation θ for a normalization factor. There is little need for normalizing a multiplier set if θ is in the neighbourhood of 1. However, to be coherent in our approach we normalize all multiplier sets. The multiplicative variant (17.6) is used for these normalizations.

All multipliers in the no initialization set are equal to 1, so this set is normalized. The formula for a quota ratio initialization multiplier is $\mathring{\lambda}_i = \frac{R_i}{q_i}$. Because this ratio is close to 1 for most constituencies, the normalization factor for a quota ratio initialization set, denoted θ_Q, is usually in the neighbourhood of 1.

As described in the previous section, the magnitude of the multipliers in an apportionment initialization set depends on the divisor method. When $CP_{0,01}$ is used, all multipliers are usually below 1, while HA results in a large portion of multipliers above 1. This influences the normalization factor, denoted θ_A. The following relationship exists between the constant parametric divisor methods and θ_A: For a given election situation and with given τ and $\mathring{\kappa}$, θ_A is a strictly decreasing function of the parameter t. To verify this statement we focus on the situation within an arbitrary constituency: Compared with the current situation a higher t will lead to a lower target quotient, which from equation (11.26) results in a higher constituency multiplier. A higher multiplier increases the right hand side of (17.6), which finally leads to a lower θ_A. From calculations of θ_A for the 23 data sets in the main matrix bias test we can say the following about the relationship between the divisor method used and the size of θ_A: θ_A is much larger than 1 for $CP_{0,01}$, while it is smaller than 1 for HA (CP_1). When MF ($CP_{0,5}$) is used, θ_A is usually in the neighbourhood of 1.

Now to the normalization of a suitable multiplier set. We let θ_S denote the normalization factor for such a set. Due to the starting position of the iterative apportionment algorithm, an apportionment initialization multiplier set, and the possibility of sharp changes for some of the multipliers during the solution process, θ_S may deviate substantially from 1.

Normalization of the multiplier sets opens for a standardization of the components. We write the standardized components in italics. The *bound*, *constituency*, and *matrix* components are calculated as prescribed by (17.2), (17.3), and (17.4) respectively, but based on normalized multiplier sets. To illustrate the relationship between the components defined in section 17.2 and their standardized counterparts, we deduce the formula for the *constituency* component:

(17.8) $$Constituency = \frac{\theta_A \cdot \lambda_i \text{ (Apportionment)}}{\theta_Q \cdot \lambda_i \text{ (Quota ratio)}} = \frac{\theta_A}{\theta_Q} \cdot \text{Constituency}$$

Similarly, we deduce the relationships:

(17.9) $$Bound = \theta_Q \cdot \text{Bound} \quad \text{and} \quad Matrix = \frac{\theta_S}{\theta_A} \cdot \text{Matrix}$$

In the previous section we could discover a multiplier which had not been adjusted during the solution process through a matrix component equal to 1. Discovering non-adjusted multipliers is a little bit more difficult here because one has to look out for *matrix* components equal to $\frac{\theta_S}{\theta_A}$ instead.

The decomposition of a normalized suitable multiplier set, denoted *Suitable*, is similar to the decomposition in (17.5):

(17.10) $$\lambda_i\,(Suitable) = Bound \cdot Constituency \cdot Matrix$$

17.4 Numerical example

In this section we present a small numerical example which illustrates the decomposition process. Table 17.1 below presents the election situation which is the basis for the example. We recognize the vote matrix from the examples in chapter 13 and section 16.3. The bound vector here is SD-HA, i.e. SD and HA have been used to determine constituency and party bounds respectively. We have decided to use MF as matrix apportionment method in the present situation.

Table 17.1

ij	B	D	G	Votes	Seats
Rev	9743	27736	9440	46919	15
NoE	6015	4606	2741	13362	4
Aul	3668	1760	1257	6685	3
Votes	19426	34102	13438	66966	
Seats	6	12	4		22

3 × 3 matrix apportionment problem.

The first task is to determine the 4 multiplier sets. How the no initialization set, the quota ratio initialization set, and the apportionment initialization set are determined should be clear from the description in section 11.3. These sets are presented under the heading "Original" in Table 17.2 below. The suitable set is found as follows: From the assignment determined by the apportionment initialization multipliers the matrix apportionment problem is solved using ρ-effect selection. The adjustments during the solution process are as follows: 1st iteration: Downadjustment to $\lambda_{Rev} = 0{,}785$, 2nd iteration: Upadjustment to $\lambda_{Aul} = 1{,}229$, 3rd iteration: Downadjustment to $\lambda_{NoE} = 0{,}747$, 4th iteration: Upadjustment to $\lambda_{Rev} = 0{,}808$, and finally 5th iteration: Downadjustment to $\lambda_{Aul} = 1{,}217$.

Table 17.2

	No	Quota ratio		Apportionment		Suitable	
	Original	Original	Normalized	Original	Normalized	Original	Normalized
λ Rev	1,000	0,973	0,913	0,919	0,967	0,808	0,895
λ NoE	1,000	0,911	0,855	0,829	0,872	0,747	0,828
λ Aul	1,000	1,366	1,281	1,127	1,186	1,217	1,348
Normalization factor			0,938		1,052		1,108

Original and normalized versions of the 4 multiplier sets.

In addition to the original multiplier sets, Table 17.2 includes normalization factors and normalized multipliers sets. Let us illustrate the normalization process by going through the calculations for the quota ratio initialization set: First we determine the normalization factor θ_Q by putting the original multipliers into formula (17.6) and solving: $\theta_Q = (0{,}973 \cdot 0{,}911 \cdot 1{,}366)^{-\frac{1}{3}} \approx 0{,}938$. Thereafter we find the normalized multipliers by multipling the original multipliers by the normalization factor, e.g. $\lambda_{Aul} = 1{,}366 \cdot 0{,}938 \approx 1{,}281$.

From the multiplier sets in Table 17.2 we calculate the components. The result of these calculations is presented in Table 17.3 below. We demonstrate the calculations by showing how the constituency components for NoE are found:

$$\text{Constituency (NoE)} = \frac{0{,}829}{0{,}911} \approx 0{,}910 \qquad \textit{Constituency} \text{ (NoE)} = \frac{0{,}872}{0{,}855} \approx 1{,}020$$

The last component can also be found with the help of (17.8):

$$\textit{Constituency} \text{ (NoE)} = \text{Constituency (NoE)} \cdot \frac{\theta_A}{\theta_Q} = 0{,}910 \cdot \frac{1{,}052}{0{,}938} \approx 1{,}020$$

Table 17.3

	between original multiplier sets			between normalized multiplier sets		
	Bound	Constituency	Matrix	*Bound*	*Constituency*	*Matrix*
Rev	0,973	0,945	0,879	0,913	1,059	0,926
NoE	0,911	0,910	0,901	0,855	1,020	0,950
Aul	1,366	0,825	1,080	1,281	0,926	1,137

Components calculated from the multiplier sets in Table 17.2.

The following information about the number of iterations with the different initialization methods is of interest below: With the election situation in Table 17.1, MF as matrix apportionment method, and ρ-effect selection, it took 6 iterations to solve the problem from the no initialization multiplier set, only 1 iteration from the quota ratio initialization set, and 5 iterations from the apportionment initialization set.

We round off this section with a brief analysis of the standardized components: The *bound* component tells us that the constituency bounds favour Austurland. This is not surprising since SD, which favours small constituencies, was used to determine these bounds. The *constituency* component pulls all constituency multipliers closer to unity. This action tells us that based on the election situations within the constituencies the quota ratio initialization multipliers are too far away from unity. The *matrix* component, which shows the necessary adjustment to reach a suitable multiplier set, more than reverses the effect of the *constituency* component. Since the *constituency* and *matrix* components pull in opposite directions and it only took 1 iteration to solve the matrix apportionment problem

after quota ratio initialization, our conclusion is that quota ratio initialization reads the problem at hand better than apportionment initialization.

Chapter 18: Three dimensions

In this chapter we extend the matrix apportionment problem to three dimensions. The first section presents the three-dimensional apportionment problem. In section 18.2 we show that the LP-formulation of the three-dimensional apportionment problem does not always yield an integer solution. The final section discusses whether there is a need for a third dimension in the apportionment problem.

18.1 The three-dimensional apportionment problem

Regarding the sets of constituencies M and parties N, we maintain our assumptions from section 9.1 We introduce the set O for the third dimension. O has a total of o members, which we call levels. We use the index k for the levels, so $k \in O = \{1, 2, ..., o\}$ where o is a positive integer. The situation where $o = 1$ is nothing but the (two-dimensional) matrix apportionment problem, so o must be larger than 1, i.e. $o \geq 2$, to move into new territory.

The vote (population) matrix $\mathbf{p} = (p_{ijk})$ is now three-dimensional, where p_{ijk} represents the votes cast for a candidate from level k of party j in constituency i. Both $\mathbf{p}$ and the apportionment matrix $\mathbf{a} = (a_{ijk})$ are $m \times n \times o$ matrices with non-negative elements, but while $\mathbf{p}$'s elements are real numbers, $\mathbf{a}$'s elements are integer. Vote and apportionment sums plus constituency and party bounds are denoted the same way as before. Below we introduce the new bounds. As before, we let lower case letters denote lower bounds and capital letters denote upper bounds. Furthermore, we assume that all bounds are integer.

We start with the bounds for the level sums. The level bounds are gathered in the vectors $\mathbf{g} = (g_k)$ and $\mathbf{G} = (G_k)$, where

(18.1) $$\mathbf{g} \geq 0 \quad \text{and} \quad \mathbf{G} > 0$$

The apportionment to level k must be within the following bounds:

(18.2) $$g_k \leq a_{MNk} \leq G_k \qquad \forall\, k$$

By summing over all levels we find the following relationship:

(18.3) $$g_O = \sum_{k \in O} g_k \leq h \leq \sum_{k \in O} G_k = G_O$$

The apportionment sums we get when summing over two indices in the three-dimensional problem, i.e. a_{iNO}, a_{MjO}, and a_{MNk}, are called **axial sums**.

We may also introduce bounds for the sums a_{ijO}, a_{iNk}, and a_{Mjk}. These sums, which are found by summing over one index, are called **planar sums**. We denote the planar sum bounds u_{ij}, U_{ij}, v_{ik}, V_{ik}, w_{jk}, and W_{jk} respectively. Thus, like we have done in the notation of g_k and G_k above and r_i, R_i, c_j, and C_j earlier, we omit the subscript(s) for the set(s) we have summed over. Later in this section we also omit subscripts for vote sums, i.e. p_{MjO} is written as p_j, p_{iNk} as p_{ik} etc. It is possible to omit subscripts for apportionment sums too, but we do not do this.

We gather the parameters for the three-dimensional problem in the **bound vector**: $\sigma = (\mathbf{u}, \mathbf{U}, \mathbf{v}, \mathbf{V}, \mathbf{w}, \mathbf{W}, \mathbf{c}, \mathbf{C}, \mathbf{r}, \mathbf{R}, \mathbf{g}, \mathbf{G}, h)$ where $\mathbf{u}$ and $\mathbf{U}$, $\mathbf{v}$ and $\mathbf{V}$, and $\mathbf{w}$ and $\mathbf{W}$ are matrices of size $m \times n$, $m \times o$, and $n \times o$ respectively. σ is integer in all its components. The problem is equality constrained if all lower bounds coincide with the corresponding upper bounds.

To formulate the problem as a constrained optimization problem we once again introduce a 0/1 apportionment (seat) variable, here denoted a_{ijkl}. Furthermore, we here let S denote the set of cells for which $p_{ijk} > 0$, while $\bar{S}$ denotes the set of cells

for which $p_{ijk} = 0$. Then we are ready for the general formulation of the **three-dimensional apportionment problem**:

(18.4) $$\max \sum_{(i,j,k)\in S} \sum_{l\in H} [\ln(\frac{p_{ijk}}{d_l}) \cdot a_{ijkl}]$$

subject to the constraints

(18.5) $$\sum_{i\in M}\sum_{j\in N}\sum_{k\in O}\sum_{l\in H} a_{ijkl} = a_{MNOH} = h$$

(18.6) $$\sum_{j\in N}\sum_{k\in O}\sum_{l\in H} a_{ijkl} = a_{iNOH} \geq r_i \qquad \forall\, i$$

(18.7) $$\sum_{j\in N}\sum_{k\in O}\sum_{l\in H} a_{ijkl} = a_{iNOH} \leq R_i \qquad \forall\, i$$

(18.8) $$\sum_{i\in M}\sum_{k\in O}\sum_{l\in H} a_{ijkl} = a_{MjOH} \geq c_j \qquad \forall\, j$$

(18.9) $$\sum_{i\in M}\sum_{k\in O}\sum_{l\in H} a_{ijkl} = a_{MjOH} \leq C_j \qquad \forall\, j$$

(18.10) $$\sum_{i\in M}\sum_{j\in N}\sum_{l\in H} a_{ijkl} = a_{MNkH} \geq g_k \qquad \forall\, k$$

(18.11) $$\sum_{i\in M}\sum_{j\in N}\sum_{l\in H} a_{ijkl} = a_{MNkH} \leq G_k \qquad \forall\, k$$

(18.12) $$\sum_{k\in O}\sum_{l\in H} a_{ijkl} = a_{ijOH} \geq u_{ij} \qquad \forall\, i,j$$

(18.13) $$\sum_{k\in O}\sum_{l\in H} a_{ijkl} = a_{ijOH} \leq U_{ij} \qquad \forall\, i,j$$

(18.14) $$\sum_{j\in N}\sum_{l\in H} a_{ijkl} = a_{iNkH} \geq v_{ik} \qquad \forall\, i,k$$

(18.15) $$\sum_{j\in N}\sum_{l\in H} a_{ijkl} = a_{iNkH} \leq V_{ik} \qquad \forall\, i,k$$

(18.16) $$\sum_{i\in M}\sum_{l\in H} a_{ijkl} = a_{MjkH} \geq w_{jk} \qquad \forall\, j,k$$

(18.17) $$\sum_{i\in M}\sum_{l\in H} a_{ijkl} = a_{MjkH} \leq W_{jk} \qquad \forall\, j,k$$

(18.18) $a_{ijkl} = 0$ or 1 $\qquad \forall\, (i,j,k) \in S, l$

(18.19) $a_{ijkl} = 0$ $\qquad \forall\, (i,j,k) \in \bar{S}, l$

(18.6) - (18.11) are axial sum constraints, while (18.12) - (18.17) are planar sum constraints. It is a matter for consideration whether all these constraints shall be included or not. A lower bound constraint is made inactive by setting the bound equal to zero, e.g. $w_{jk} = 0$, while an upper bound constraint is made inactive by setting the bound equal to the house size, e.g. $W_{jk} = h$.

The question which now arises is: How should the bounds in the three-dimensional apportionment problem be determined? For the (two-dimensional) matrix apportionment problem we recommended the use of apportionment methods for this task. We recommend the same approach here. However, the larger number of bounds makes the determination process more cumbersome. Below we review the determination process for the three-dimensional equality constrained problem. Since the upper and lower bounds are identical, we use the notation for upper bounds, i.e. capital letters, in this description. Furthermore, we take the house size h as given.

We start by determining the axial bounds **R**, **C**, and **G**. These bounds are determined by vector apportionments based on the respective vote totals, i.e. based on the p_is, the p_js, and the p_ks respectively. We will always be able to determine **R**, **C**, and **G** this way. The next task is the determination of planar bounds. A natural way of determining the bound matrices **U**, **V**, and **W** is to carry out three (two-dimensional) matrix apportionments. The matrix apportionment to determine **U** is based on the p_{ij}s with **R** and **C** as bounds. **V** and **W** are determined similarly, based on the p_{ik}s and the p_{jk}s respectively. As explained in section 9.7 and illustrated by Example 9.2 and Example 9.3, a matrix apportionment problem may not have a feasible solution. Thus, it may be impossible to determine **U**, **V**, and/or **W** by matrix apportionments. When the prescribed matrix apportionments are feasible, we have the following relationships between the planar bounds **U**, **V**, and **W** and the axial bounds **R**, **C**, and **G**:

$$\sum_{j \in N} U_{ij} = \sum_{k \in O} V_{ik} = R_i \qquad \forall\, i \tag{18.20}$$

$$\sum_{i \in M} U_{ij} = \sum_{k \in O} W_{jk} = C_j \qquad \forall\, j \tag{18.21}$$

$$\sum_{i \in M} V_{ik} = \sum_{j \in N} W_{jk} = G_k \qquad \forall\, k \tag{18.22}$$

(18.20) - (18.22) are necessary conditions for the existence of a feasible solution to the equality constrained three-dimensional apportionment problem. However, these conditions are far from sufficient, see [V] for a survey.

18.2 Example of a non-integer LP-solution

Due to constraint (18.18), the problem defined by (18.4) - (18.19) is a 0/1 integer programming problem. If we reformulate (18.18) as:

$$a_{ijkl} \geq 0 \qquad \forall\, (i,j,k) \in S, l \tag{18.23}$$

$$a_{ijkl} \leq 1 \qquad \forall\, (i,j,k) \in S, l \tag{18.24}$$

the problem is formulated as a LP-problem. With the LP-formulation of the (two-dimesional) matrix apportionment problem we were guaranteed an optimal solution which was integer provided that all components of the bound vector σ were integer. We have no such guarantee with the LP-formulation of the three-dimensional apportionment problem, as the following example shows:

Example 18.1

We face a 3 × 3 × 3 election situation, where the three constituencies are named 1, 2, and 3, the three parties A, B, and C, and the three levels I, II, and III. The vote matrix $\mathbf{p} = (p_{ijk})$, where $\psi > 1$, is presented in the three tables below. Each table shows the votes for one of the levels:

Table 18.1

I	A	B	C
1	ψ	1	1
2	1	ψ	1
3	1	1	1

Vote submatrix (p_{ijI})

Table 18.2

II	A	B	C
1	1	1	1
2	ψ	1	1
3	1	1	ψ

Vote submatrix (p_{ijII})

Table 18.3

III	A	B	C
1	1	1	ψ
2	1	1	1
3	1	ψ	1

Vote submatrix (p_{ijIII})

We assume that a strict divisor series d_l is being used. Our task is to find the optimal solution to the following LP-formulation of the three-dimensional axial apportionment problem:

$$\max \sum_{i=1}^{3}\sum_{j=A}^{C}\sum_{k=I}^{III}\sum_{l\in H} [\ln(\frac{p_{ijk}}{d_l}) \cdot a_{ijkl}] \tag{18.25}$$

subject to the constraints

$$\sum_{i=1}^{3}\sum_{j=A}^{C}\sum_{k=I}^{III}\sum_{l\in H} a_{ijkl} = a_{MNOH} = 3 \tag{18.26}$$

$$\sum_{j=A}^{C}\sum_{k=I}^{III}\sum_{l\in H} a_{ijkl} = a_{iNOH} = 1 \qquad \forall\, i \tag{18.27}$$

$$\sum_{i=1}^{3}\sum_{k=I}^{III}\sum_{l\in H} a_{ijkl} = a_{MjOH} = 1 \qquad \forall\, j \tag{18.28}$$

$$\sum_{i=1}^{3}\sum_{j=A}^{C}\sum_{l\in H} a_{ijkl} = a_{MNkH} = 1 \qquad \forall\, k \tag{18.29}$$

$$a_{ijkl} \geq 0 \qquad \forall\, i,j,k,l \tag{18.30}$$

$$a_{ijkl} \leq 1 \qquad \forall\, i,j,k,l \tag{18.31}$$

We observe that the right hand sides of (18.27) - (18.29) are proportional to the constituency, party, and level vote totals respectively. The axial bounds of 1 together with the strict divisor series d_l guarantee that no second quotient q_{ijk2} will be apportioned any (part of a) seat. In fact, with a 0/1 requirement instead of (18.30) - (18.31), the problem would have been an axial assignment problem in three dimensions.

We concluded above that only first quotients will be apportioned seats, so we omit the *l* subscript in the description below. Our search for the optimal solution to the LP-problem starts by constructing feasible integer solutions. We begin with Table 18.1 and assign seats greedily: Because $\psi > 1$, one of the two cells for which $p_{ijI} = \psi$ should be assigned a seat. Let us set $a_{1AI} = 1$. Then the capacities for constituency 1, party A, and level I are saturated. We move on to the vote table for the next level. Like the first table, Table 18.2 contains two ψ elements. Since the capacity for party A already is saturated, we set $a_{3CII} = 1$, which saturates the capacities for constituency 3, party C and level II. This leaves us with $a_{2BIII} = 1$ as the only available assignment in Table 18.3. Unfortunately $p_{2BIII} = 1$, so the value of the objective function with the chosen assignment is $2 \cdot \ln(\psi) - 3 \cdot \ln(d_1)$. If we instead had started out by setting $a_{2BI} = 1$ and followed up by setting $a_{3CII} = 1$, we would have been forced to let $a_{1AIII} = 1$ resulting in the same objective function value. The last possibility in Table 18.1 is to start by assigning a seat to a cell for which $p_{ijI} = 1$. However, the best objective function value one can hope for then is $2 \cdot \ln(\psi) - 3 \cdot \ln(d_1)$. Thus, $2 \cdot \ln(\psi) - 3 \cdot \ln(d_1)$ is the highest attainable objective function value for an integer solution. It is possible to improve on this value by allowing non-integer values for the variables. The solution found by letting all variables representing cells with a population of ψ be equal to $\frac{1}{2}$, i.e. $a_{1AI} = a_{2BI} = a_{2AII} = a_{3CII} = a_{1CIII} = a_{3BIII} = \frac{1}{2}$, results in an objective function value of $3 \cdot \ln(\psi) - 3 \cdot \ln(d_1)$. Clearly, this is the optimal solution to the LP-problem.

18.3 Motivation for introducing a third dimension

Is there a need for a third dimension in the matrix apportionment problem? In this section we evaluate sex as a possible third dimension. We also review some other possibilities regarding the third dimension.

The first characteristic which came to mind as possibly justifying an own dimension was sex. Norway has a law (Law of equal status) with the intention of

securing women's representation in public committees. This law states that public committees, with the exception of elected ones, shall have members from both sexes and that the representation of the least represented sex shall be at least 40% if a committee has 4 members or more. It is possible to deviate from this requirement if it is highly unreasonable for the actual committee [Likestillingsloven 9 juni. Nr. 45. 1978 § 21]. What about extending this law to include representation in elected assemblies too? We will deal with this question later on, but let us start by studying the development of the Storting's **women percentage** during the last four terms: In the general election in 1985, 54 out of the 157 elected representatives to the Storting were women, i.e. a percentage of $\frac{54}{157} \approx 34\%$, in 1989 the percentage increased to $\frac{59}{165} \approx 36\%$, in 1993 it reached $\frac{65}{165} \approx 39\%$, while it dropped to $\frac{60}{165} \approx 36\%$ in the most recent election in 1997. Thus, the Storting's women percentage has never been above 40%. In the light of the fact that over 50% of the eligible voters are women, the referred women percentages are not impressive.

Let us now look at how lower bounds for the representation of the sexes can be included in the apportionment problem. A possibility is to let the apportionment problem become three-dimensional and operate with separate party lists for men and women in every constituency. Then the lower bound constraint for the representation of each sex can be written as:

$$(18.32) \qquad \sum_{i \in M} \sum_{j \in N} \sum_{l \in H} a_{ijkl} = a_{MNkH} \geq \nu \cdot h \quad \forall\, k \qquad \text{where } \nu \in [0, \tfrac{1}{2}]$$

ν can be interpreted as a lower bound percentage. Notice that (18.32) is (18.10) with $g_k = \nu \cdot h$. Since $o = 2$ with sex as the third dimension, the lower bound for one sex determines an upper bound for the opposite sex: $a_{MNkH} \leq (1 - \nu) \cdot h$.

An alternative to the approach described above is to handle the two lower bound sex constraints within the two-dimensional framework. This is accomplished by operating with the same party lists as in the (two-dimensional) matrix apportionment problem. In addition, one must keep track of which quotients belong to female (male) candidates. The two lower bound sex constraints have the same

right hand side as (18.32), and they are constructed by summing over the seat variables a_{ijl} belonging to female and male candidates respectively. Notice that these two constraints will break with the transportation type structure of the (two-dimensional) matrix apportionment problem. Moreover, if one of them becomes active, the solution process will be more complicated than described in chapter 12. The advantage of the approach described in this paragraph is that party lists can be nominated the same way they are today.

In this and the next paragraph we present examples from Norway of connections between sex and party. The first example shows how two of the parties deal with the representation of the sexes: Both A (Labour Party) and SV (Socialist Left Party) nominate their party lists such that among every second person on the list there are one female and one male candidate. By this description we mean that if there is a man on top of the list, a woman is in 2nd place, and if there is a woman in 3rd place, a man is in 4th place etc. This sounds promising for these parties' women percentages, but the additional piece of information that both parties had a woman in 1st place in only 7 out of 19 constituencies, i.e. 37%, in the general election in 1993 reduces the expectations. What is interesting from an apportionment perspective is that while the women percentage for A became as high as $\frac{33}{67} \approx 49\%$, SV ended up with $\frac{4}{13} \approx 31\%$. The reason for SV's low percentage is its small size, which makes it very important to be on top of the party list. For a small party the battle in nearly every constituency is for its first seat, while a large party like A battles for its seat no. three, four etc. Party nominations are carried out separately for each constituency. In order to obtain a certain women percentage at the national level, a party must coordinate the nominations in the different constituencies.

We continue with some other data regarding sex and party preference. Table 18.4 below presents the following data for each party which won seat(s) in the Storting in the general election in 1993: The women percentage among the party's voters, the women percentage in the parliamentary party group, plus the ratio between these two percentages. To estimate the women percentages among the different parties' voters, we utilize data from a sample survey of the 1993 general election

[Statistisk Sentralbyrå - Stortingsvalget 1993]. This survey is based on less than two thousand answers, but the election result among the respondents did not deviate much from the national election result. The data of interest to us are the percentage support for the parties among male and female voters respectively. These percentages are given with no decimals, which result in rough estimates of the women percentages, as illustrated below. Although over 50% of the eligible voters were women and the participation rate was slightly higher among women than among men, we assume that 50% of the votes were cast by men when estimating the women percentage among a party's voters. The following example illustrates the estimation of "% among voters": According to the survey, FrP (Progress Party) had 6% support among male and 3% support among female voters. These figures result in an estimated women percentage of $\frac{3\%}{3\% + 6\%} \approx 33\%$. This is a rough estimate; given the precision of the survey figures, the support among female and male voters might have been 2,6% and 6,4% respectively. With these figures the correct estimate of FrP's women percentage would have been $\frac{2,6\%}{9,0\%} \approx 29\%$. "% in party group" is calculated from statistics about the elected representatives to the Storting, while "Ratio" is calculated as $\frac{\%\text{ among voters}}{\%\text{ in party group}}$. A ratio near 1,0 tells us that the sex composition of the party group harmonizes well with the sex composition of the people who voted for the party. Moreover, a ratio higher than 1,0 shows that there are too few women in the party group compared with the composition of the party's voters.

Table 18.4

Parties	A	FrP	H	KrF	RV	SP	SV	V
% among voters	50%	33%	44%	60%	67%	50%	60%	57%
% in party group	49%	10%	29%	38%	0%	44%	31%	0%
Ratio	1,0	3,3	1,5	1,6	∞	1,1	1,9	∞

Women percentages for the Norwegian parties in the 1993 general election.

With the exception of RV (Red Electoral Alliance) and V (Liberal party), which party groups consisted of only one person each, the parties had from 10 (FrP) to 63 representatives (A) in the Storting. Considering the political views of the parties, the two opposites regarding the achieved ratios must be the two parties

from the preceding example: A was presumably well satisfied with its ratio, while SV must have viewed its ratio as unsatisfactory.

Now back to the law of equal status. Why does not this law include elections? We believe that the reason is a predominant view that political parties through their nomination of candidates should deal with such matters as the candidates' sex etc. A lower bound in the spirit of (18.32) would interfere with the priorities of the parties if it becomes an active constraint.

It is possible to include other characteristics than sex, like race, religion, age etc, in the apportionment problem. However, a direct vote on race and/or religion is presumably neither desirable nor politically acceptable. It should not be necessary either because in countries where race or religion are issues which divide the nation, the formation of parties usually follows these dividing lines. From the Nordic countries we have examples of formation of parties based on age and sex respectively: The "Pensioners' Party" in Norway is represented in several county and municipality councils. In Iceland the "Women's Alliance" was represented in the Althing from 1983 to 1999. Its support in the last election it participated in was 5%, but the support had earlier been as high as 10%. We end this paragraph by rejecting another possibility regarding the third dimension: Time is sometimes introduced as an own dimension in transportation type problems, but we do not find it applicable in the apportionment context.

It is time to conclude the discussion regarding introduction of own dimensions for characteristics like sex etc. The basic assumption in the (two-dimensional) matrix apportionment problem is that every voter wants his/her own constituency and the party he/she prefers to be represented by as many representatives as possible. We find it less reasonable to assume that the sex or any other characteristic of a representative, and which is not covered by the party choice, is of the same importance to the voter. Furthermore, the point of introducing a new dimension must be that one wants the apportionment to be proportional for axial and/or planar sums involving this dimension. Since we do not see the need for proportional representation in parliament for any of the above-mentioned

characteristics, our view is that none of them justify an own dimension in the apportionment problem.

We end this review of the three-dimensional apportionment problem by presenting two applications we find interesting: As explained in section 15.5, divisor methods may be employed for some controlled rounding problems. Some of these problems are of dimension three or higher, see [K&al] (page 761), so this is a field where divisor method formulations akin to that in section 18.1 should be of interest.

During the work with the material above, we came up with the following idea for a three-dimensional apportionment problem with both axial and planar sum constraints: The members of the Storting are divided in different committees; Defence Committee, Judiciary Committee etc. Today the composition of these committees is determined by the representatives themselves after the election. Why do not we let the voters have a direct influence on the composition of the committees? The possibility of voting for both party and committee would be an opportunity for the voters to express why they prefer their party. Moreover, the committee ballot would show which topics are important for voters in the different regions of Norway. Although the political power of the committees is limited, the committee ballot would serve as an opinion poll, and everybody knows how politicians react to opinion polls. An argument against the proposed voting scheme is that it will place a heavier burden on the voters. Another drawback is that the party composition in the different committes may be quite skew compared with the composition of the Storting. Today the composition of each committee approximately reflects the composition of the Storting. To secure a suitable number of representatives in the different committees with the proposed voting scheme, one should operate with predetermined bounds for the committee sizes.

References

[Ba] Bacharach, Michael (1970). *Biproportional Matrices and Input-Output Change.* Cambridge University Press, Cambridge, England.

[B] Balinski, Michel (1993). "The Problem with Apportionment," *Journal of the Operations Research Society of Japan* **36**, 134-148.

[B&D-1] Balinski, M. L. and Demange, G. (1989). "An Axiomatic Approach to Proportionality between Matrices," *Mathematics of Operations Research* **14**, 700-719.

[B&D-2] Balinski, M. L. and Demange, G. (1989). "Algorithms for Proportional Matrices in Reals and Integers," *Mathematical Programming* **45**, 193-210.

[B&S] Bigelow, James H. and Shapiro, Norman Z. (1977). "A Scaling Theorem and Algorithms," *SIAM Journal of Applied Mathematics* **33**, 348-352.

[B&Y] Balinski, Michel L. and Young, H. Peyton (1982). *Fair Representation: Meeting the Ideal of One Man, One Vote.* Yale University Press, New Haven, CT.

[C] Cox, Lawrence H. (1987). "A Constructive Procedure for Unbiased Controlled Rounding," *Journal of the American Statistical Association* **82**, 520-524.

[C&E] Cox, Lawrence H. and Ernst, Lawrence R. (1982). "Controlled Rounding," *INFOR* **20**, 423-432.

[C&C&E] Causey, Beverley D., Cox, Lawrence H., and Ernst, Lawrence R. (1985). Applications of Transportation Theory to Statistical Problems, *Journal of the American Statistical Association* **80**, 903-909.

[D&C] Dunn, Olive Jean and Clark, Virginia A. (1974). *Applied statistics: Analysis of Variance and Regression.* Wiley, New York, NY.

[E&G&S] Eppen, G. D., Gould, F. J., and Schmidt, C. P. (1991). *Introductory Management Science.* 3rd edition, Prentice Hall, Englewood Cliffs, NJ.

[E] Ernst, Lawrence R. (1994). "Apportionment Methods for the House of Representatives and the Court Challenges," *Management Science* **40**, 1207-1227.

[G&N] Garfinkel, R. S. and Nemhauser G. L. (1970). "Optimal Political Districting by Implicit Enumeration Techniques," *Management Science* **16**, B495-B508.

[G]* Grønvik, Gunnvald (1989). "Fra stemmer til mandater," *Tidsskrift for samfunnsforskning*, 239-262.

[H&J] Helgason, Thorkell and Jörnsten, Kurt (1994). *Entropy of Proportional Matrix Apportionments*. Working paper 4/94, Institute of Finance and Management Science, Norwegian School of Economics and Business Administration, Bergen, Norway.

[Hu] Huntington, E. V. (1928). "The Apportionment of Representatives in Congress," *Transactions of the American Mathematical Society* **30**, 85-110.

[H-S]* Hylland, Aanund (1988). *Valgordningen ved Stortingsvalg*. Working paper 1988/32, Norwegian School of Management, Sandvika, Norway.

[H-A] Hylland, Aanund (1990). *Allotment Methods: Procedures for Proportional Distribution of Indivisible Entities*. Working Paper 1990/11, Norwegian School of Management, Sandvika, Norway.

[K&al] Kelly, James P., Golden, Bruce L., Assad, Arjang A. and Baker, Edward K. (1990). "Controlled Rounding of Tabular Data," *Operations Research* **38**, 760-772.

[L] Sainte-Laguë, André (1910). "La représentation proportionelle et la méthode des moindres carrées," *Comptes rendus des séances de l'Académie des sciences* **151**, 377-378, Paris, France.

[L&G] Lijphart, Arend and Gibberd, Robert W. (1977). "Thresholds and Payoffs in List Systems of Proportional Representation," *European Journal of Political Research* **5**, 219-244.

[Li]* Lillstøl, Jostein (1986). *Sannsynlighetsregning og statistikk med anvendelser*. 3. utgave, Bedriftsøkonomens Forlag, Norway.

[O] Oyama, Tatsuo (1991). "On a Parametric Divisor Method for the Apportionment Problem," *Journal of the Operations Research Society of Japan* **34**, 187-221.

[R] Rae, Douglas W. (1971). *The Political Consequences of Electoral Laws*. Yale University Press, New Haven, CT.

[Sp] Späth, Helmuth (1985). *Cluster Dissection and Analysis: Theory, FORTRAN Programs, Examples*. Ellis Horwood, Chichester, England.

[S]* Syversten, Bjørne-Dyre H. (1992). *Mandatfordelinger i én og to dimensjoner*. HAS-thesis, Norwegian School of Economics and Business Administration, Bergen, Norway.

[V] Vlach, Milan (1986). "Conditions for the Existence of Solutions of the Three-Dimensional Planar Transportation Problem," *Discrete Applied Mathematics* **13**, 61-78.

* = In Norwegian

Appendices

Appendix 1:

Divisions

and

Results of vector bias test

Description of Division tables

These tables show the following for the given number of groups c:

- How many constituencies there are in each of the 13 countries utilized
- Number, Cluster, Quota, and Size divisions

We always order the constituencies within a country in descending order based on population. The intervals X - Y given in the tables show the first member X and the last member Y of the group, where X and Y are the Xth and Yth largest constituency in the country respectively.

Description of Vector bias tables

These tables show the vector bias data for the comparison of group V and W when there are c groups. $V < W \leq c$, where $c = 2$, 3, or 4.

Vector bias data are shown for all combinations of the 6 vector apportionment methods; SD, HM, EP, MF, HA, and LF, and the 4 division methods; Cluster, Quota, Number, and Size. The formulas for $\varepsilon(V,W)$ and T are presented in section 5.8.

Division in 2 groups

Country	# of constituencies	Division	Group 1	Group 2
Japan	130	Number	1 – 65	66 – 130
		Cluster	1 – 39	40 – 130
		Quota	1 – 43	44 – 130
		Size	1 – 38	39 – 130
Thailand	71	Number	1 – 35	36 – 71
		Cluster	1 – 14	15 – 71
		Quota	1 – 17	18 – 71
		Size	1 – 5	6 – 71
Greece	56	Number	1 – 28	29 – 56
		Cluster	1 – 2	3 – 56
		Quota	1 – 13	14 – 56
		Size	1 – 2	3 – 56
USA	50	Number	1 – 25	26 – 50
		Cluster	1 – 7	8 – 50
		Quota	1 – 8	9 – 50
		Size	1 – 3	4 – 50
Sweden	28	Number	1 – 14	15 – 28
		Cluster	1 – 2	3 – 28
		Quota	1 – 10	11 – 28
		Size	1 – 3	4 – 28
Switzerland	26	Number	1 – 13	14 – 26
		Cluster	1 – 3	4 – 26
		Quota	1 – 4	5 – 26
		Size	1 – 3	4 – 26
Norway	19	Number	1 – 9	10 – 19
		Cluster	1 – 4	5 – 19
		Quota	1 – 6	7 – 19
		Size	1 – 8	9 – 19
Denmark	17	Number	1 – 8	9 – 17
		Cluster	1 – 4	5 – 17
		Quota	1 – 5	6 – 17
		Size	1 – 6	7 – 17
Germany	16	Number	1 – 8	9 – 16
		Cluster	1 – 4	5 – 16
		Quota	1 – 3	4 – 16
		Size	1 – 3	4 – 16
Finland	15	Number	1 – 7	8 – 15
		Cluster	1 – 6	7 – 15
		Quota	1 – 5	6 – 15
		Size	1 – 6	7 – 15
Canada	12	Number	1 – 6	7 – 12
		Cluster	1 – 2	3 – 12
		Quota	1 – 2	3 – 12
		Size	1 – 2	3 – 12
Austria	9	Number	1 – 4	5 – 9
		Cluster	1 – 4	5 – 9
		Quota	1 – 3	4 – 9
		Size	1 – 4	5 – 9
Iceland	8	Number	1 – 4	5 – 8
		Cluster	1 – 2	3 – 8
		Quota	1 – 1	2 – 8
		Size	1 – 2	3 – 8

Vector bias 2 groups

ε(1,2) Bias between group 1 and 2 for c = 2 based on 13 countries

	SD		HM		EP		MF		HA		LF	
					Cluster division							
ε	-6,29%		-1,55%		-1,22%		-0,05%		5,79%		0,04%	
S	1,38%		0,79%		0,76%		0,35%		0,94%		0,21%	
T	-4,56	**	-1,98	*	-1,61		-0,16		6,15	**	0,21	
P	0,0%		3,6%		6,6%		87,9%		0,0%		83,9%	
					Quota division							
ε	-6,41%		-1,12%		-0,79%		0,47%		5,65%		0,44%	
S	1,27%		0,63%		0,56%		0,34%		0,93%		0,24%	
T	-5,06	**	-1,78		-1,41		1,38		6,06	**	1,83	
P	0,0%		5,0%		9,2%		19,4%		0,0%		9,2%	
					Number division							
ε	-8,66%		-2,82%		-2,03%		0,72%		8,95%		1,03%	
S	1,68%		1,07%		0,96%		0,45%		1,79%		0,32%	
T	-5,17	**	-2,64	*	-2,11	*	1,59		5,01	**	3,24	**
P	0,0%		1,1%		2,8%		13,9%		0,0%		0,7%	
					Size division							
ε	-6,31%		-1,77%		-1,36%		-0,27%		5,70%		-0,17%	
S	1,33%		0,76%		0,70%		0,28%		1,06%		0,22%	
T	-4,73	**	-2,33	*	-1,94	*	-0,95		5,39	**	-0,77	
P	0,0%		1,9%		3,8%		36,3%		0,0%		45,5%	

ε Estimated bias percentage (mean)
S Standard error of the estimate
T t-statistic
P P-value (probability); two-sided for MF and LF, one-sided for the other methods

** Significant at the 1%-level
* Significant at the 5%-level

Division in 3 groups

Country	# of constituencies	Division	Group 1	Group 2	Group 3
Japan	130	Number	1 – 43	44 – 86	87 – 130
		Cluster	1 – 21	22 – 73	74 – 130
		Quota	1 – 25	26 – 65	66 – 130
		Size	1 – 17	18 – 88	89 – 130
Thailand	71	Number	1 – 23	24 – 47	48 – 71
		Cluster	1 – 3	4 – 23	24 – 71
		Quota	1 – 9	10 – 28	29 – 71
		Size	1 – 3	4 – 14	15 – 71
Greece	56	Number	1 – 18	19 – 37	38 – 56
		Cluster	1 – 2	3 – 20	21 – 56
		Quota	1 – 7	8 – 22	23 – 56
		Size	1 – 1	2 – 3	4 – 56
USA	50	Number	1 – 16	17 – 33	34 – 50
		Cluster	1 – 1	2 – 8	9 – 50
		Quota	1 – 4	5 – 15	16 – 50
		Size	1 – 1	2 – 7	8 – 50
Sweden	28	Number	1 – 9	10 – 18	19 – 28
		Cluster	1 – 2	3 – 23	24 – 28
		Quota	1 – 5	6 – 15	16 – 28
		Size	1 – 2	3 – 8	9 – 28
Switzerland	26	Number	1 – 8	9 – 17	18 – 26
		Cluster	1 – 2	3 – 8	9 – 26
		Quota	1 – 2	3 – 7	8 – 26
		Size	1 – 2	3 – 5	6 – 26
Norway	19	Number	1 – 6	7 – 12	13 – 19
		Cluster	1 – 3	4 – 12	13 – 19
		Quota	1 – 3	4 – 9	10 – 19
		Size	1 – 4	5 – 13	14 – 19
Denmark	17	Number	1 – 5	6 – 11	12 – 17
		Cluster	1 – 4	5 – 15	16 – 17
		Quota	1 – 3	4 – 8	9 – 17
		Size	1 – 4	5 – 14	15 – 17
Germany	16	Number	1 – 5	6 – 10	11 – 16
		Cluster	1 – 1	2 – 4	5 – 16
		Quota	1 – 2	3 – 6	7 – 16
		Size	1 – 1	2 – 4	5 – 16
Finland	15	Number	1 – 5	6 – 10	11 – 15
		Cluster	1 – 1	2 – 9	10 – 15
		Quota	1 – 3	4 – 7	8 – 15
		Size	1 – 2	3 – 11	12 – 15
Canada	12	Number	1 – 4	5 – 8	9 – 12
		Cluster	1 – 2	3 – 4	5 – 12
		Quota	1 – 1	2 – 3	4 – 12
		Size	1 – 2	Ø	3 – 12
Austria	9	Number	1 – 3	4 – 6	7 – 9
		Cluster	1 – 4	5 – 7	8 – 9
		Quota	1 – 2	3 – 4	5 – 9
		Size	1 – 4	5 – 6	7 – 9
Iceland	8	Number	1 – 2	3 – 5	6 – 8
		Cluster	1 – 1	2 – 2	3 – 8
		Quota	1 – 1	2 – 3	4 – 8
		Size	1 – 1	2 – 2	3 – 8

Vector bias 3 groups

ε(1,2) Bias between group 1 and 2 for c = 3 based on 13 countries (12 for Size division)

	SD	HM	EP	MF	HA	LF
			Cluster division			
ε	-2,92%	-0,22%	-0,09%	0,06%	2,48%	0,01%
S	0,78%	0,55%	0,39%	0,49%	0,50%	0,39%
T	-3,75 **	-0,41	-0,24	0,12	5,01 **	0,02
P	0,1%	34,5%	40,8%	90,8%	0,0%	98,3%
			Quota division			
ε	-3,47%	-0,63%	-0,52%	-0,42%	2,11%	-0,57%
S	1,11%	0,50%	0,40%	0,35%	0,52%	0,28%
T	-3,12 **	-1,26	-1,30	-1,20	4,09 **	-2,03
P	0,4%	11,5%	10,9%	25,3%	0,1%	6,5%
			Number division			
ε	-4,08%	-0,75%	-0,60%	0,17%	4,07%	0,12%
S	0,89%	0,68%	0,66%	0,53%	0,78%	0,55%
T	-4,60 **	-1,10	-0,90	0,32	5,22 **	0,22
P	0,0%	14,7%	19,3%	75,6%	0,0%	82,6%
			Size division			
ε	-2,36%	-0,35%	0,03%	0,32%	2,27%	0,29%
S	0,73%	0,53%	0,32%	0,44%	0,68%	0,41%
T	-3,22 **	-0,65	0,11	0,73	3,35 **	0,71
P	0,4%	26,3%	45,8%	48,3%	0,3%	49,5%

ε(1,3) Bias between group 1 and 3 for c = 3 based on 13 countries (12 for Size division)

	SD	HM	EP	MF	HA	LF
			Cluster division			
ε	-9,80%	-2,58%	-1,73%	0,15%	9,32%	0,32%
S	1,68%	1,05%	0,90%	0,60%	1,37%	0,59%
T	-5,82 **	-2,47 *	-1,93 *	0,25	6,80 **	0,54
P	0,0%	1,5%	3,9%	80,6%	0,0%	59,7%
			Quota division			
ε	-9,11%	-2,12%	-1,44%	0,59%	8,53%	0,58%
S	1,95%	0,99%	0,87%	0,42%	1,58%	0,26%
T	-4,67 **	-2,14 *	-1,66	1,41	5,41 **	2,19 *
P	0,0%	2,7%	6,2%	18,5%	0,0%	4,9%
			Number division			
ε	-14,36%	-5,67%	-3,91%	1,07%	13,73%	1,95%
S	2,96%	2,34%	1,81%	0,68%	2,46%	0,84%
T	-4,85 **	-2,42 *	-2,16 *	1,56	5,57 **	2,32 *
P	0,0%	1,6%	2,6%	14,4%	0,0%	3,9%
			Size division			
ε	-9,57%	-2,36%	-1,59%	-0,06%	8,38%	-0,10%
S	1,92%	1,04%	0,83%	0,51%	1,27%	0,32%
T	-4,98 **	-2,27 *	-1,92 *	-0,13	6,62 **	-0,32
P	0,0%	2,2%	4,0%	90,1%	0,0%	75,6%

Vector Bias 3 groups

ε(2,3) Bias between group 2 and 3 for c = 3 based on 13 countries (12 for Size division)

	SD	HM	EP	MF	HA	LF
			Cluster division			
ε	-6,62%	-2,36%	-1,66%	0,06%	7,06%	0,29%
S	0,99%	0,93%	0,95%	0,87%	1,02%	0,77%
T	-6,66 **	-2,52 *	-1,74	0,07	6,94 **	0,38
P	0,0%	1,3%	5,3%	94,8%	0,0%	71,4%
			Quota division			
ε	-5,39%	-1,47%	-0,90%	1,00%	6,60%	1,14%
S	1,00%	0,68%	0,64%	0,44%	1,33%	0,33%
T	-5,40 **	-2,16 *	-1,41	2,28 *	4,95 **	3,42 **
P	0,0%	2,6%	9,2%	4,2%	0,0%	0,5%
			Number division			
ε	-9,81%	-4,87%	-3,29%	0,86%	10,14%	1,78%
S	2,45%	2,15%	1,66%	0,95%	2,26%	1,11%
T	-4,00 **	-2,26 *	-1,98 *	0,90	4,49 **	1,61
P	0,1%	2,2%	3,6%	38,4%	0,0%	13,4%
			Size division			
ε	-6,99%	-1,99%	-1,63%	-0,40%	6,28%	-0,41%
S	1,35%	0,79%	0,77%	0,61%	0,95%	0,51%
T	-5,19 **	-2,51 *	-2,11 *	-0,66	6,58 **	-0,79
P	0,0%	1,4%	2,9%	52,4%	0,0%	44,6%

ε Estimated bias percentage (mean)
S Standard error of the estimate
T t-statistic
P P-value (probability); two-sided for MF and LF, one-sided for the other methods

** Significant at the 1%-level
* Significant at the 5%-level

Division in 4 groups

Country	# of constituencies	Division	Group 1	Group 2	Group 3	Group 4
Japan	130	Number	1 – 32	33 – 64	65 – 97	98 – 130
		Cluster	1 – 17	18 – 47	48 – 92	93 – 130
		Quota	1 – 18	19 – 44	45 – 78	79 – 130
		Size	1 – 11	12 – 38	39 – 111	112 – 130
Thailand	71	Number	1 – 17	18 – 35	36 – 53	54 – 71
		Cluster	1 – 3	4 – 14	15 – 34	35 – 71
		Quota	1 – 6	7 – 17	18 – 35	36 – 71
		Size	1 – 1	2 – 5	6 – 23	24 – 71
Greece	56	Number	1 – 14	15 – 28	29 – 42	43 – 56
		Cluster	1 – 1	2 – 2	3 – 20	21 – 56
		Quota	1 – 4	5 – 14	15 – 28	29 – 56
		Size	1 – 1	2 – 2	3 – 6	7 – 56
USA	50	Number	1 – 12	13 – 24	25 – 37	38 – 50
		Cluster	1 – 1	2 – 8	9 – 27	28 – 50
		Quota	1 – 3	4 – 9	10 – 20	21 – 50
		Size	1 – 1	2 – 3	4 – 9	10 – 50
Sweden	28	Number	1 – 7	8 – 14	15 – 21	22 – 28
		Cluster	1 – 2	3 – 5	6 – 23	24 – 28
		Quota	1 – 3	4 – 9	10 – 17	18 – 28
		Size	1 – 2	3 – 3	4 – 23	24 – 28
Switzerland	26	Number	1 – 6	7 – 12	13 – 19	20 – 26
		Cluster	1 – 2	3 – 6	7 – 16	17 – 26
		Quota	1 – 2	3 – 5	6 – 11	12 – 26
		Size	1 – 2	3 – 3	4 – 7	8 – 26
Norway	19	Number	1 – 4	5 – 9	10 – 14	15 – 19
		Cluster	1 – 3	4 – 9	10 – 14	15 – 19
		Quota	1 – 2	3 – 6	7 – 11	12 – 19
		Size	1 – 3	4 – 8	9 – 16	17 – 19
Denmark	17	Number	1 – 4	5 – 8	9 – 12	13 – 17
		Cluster	1 – 4	5 – 7	8 – 15	16 – 17
		Quota	1 – 2	3 – 5	6 – 10	11 – 17
		Size	1 – 4	5 – 6	7 – 16	17 – 17
Germany	16	Number	1 – 4	5 – 8	9 – 12	13 – 16
		Cluster	1 – 1	2 – 3	4 – 6	7 – 16
		Quota	1 – 1	2 – 3	4 – 7	8 – 16
		Size	1 – 1	2 – 3	4 – 6	7 – 16
Finland	15	Number	1 – 3	4 – 7	8 – 11	12 – 15
		Cluster	1 – 1	2 – 7	8 – 14	15 – 15
		Quota	1 – 2	3 – 5	6 – 9	10 – 15
		Size	1 – 1	2 – 6	7 – 13	14 – 15
Canada	12	Number	1 – 3	4 – 6	7 – 9	10 – 12
		Cluster	1 – 1	2 – 2	3 – 4	5 – 12
		Quota	1 – 1	2 – 2	3 – 4	5 – 12
		Size	1 – 1	2 – 2	3 – 4	5 – 12
Austria	9	Number	1 – 2	3 – 4	5 – 6	7 – 9
		Cluster	1 – 2	3 – 4	5 – 7	8 – 9
		Quota	1 – 1	2 – 2	3 – 4	5 – 9
		Size	1 – 4	Ø	5 – 7	8 – 9
Iceland	8	Number	1 – 2	3 – 4	5 – 6	7 – 8
		Cluster	1 – 1	2 – 2	3 – 4	5 – 8
		Quota	1 – 1	2 – 2	3 – 4	5 – 8
		Size	1 – 1	2 – 2	3 – 3	4 – 8

Vector bias 4 groups

ε(1,2) Bias between group 1 and 2 for c = 4 based on 13 countries (12 for Size division)

	SD	HM	EP	MF	HA	LF
			Cluster division			
ε	-1,73%	-0,41%	-0,09%	0,24%	1,65%	0,20%
S	0,61%	0,46%	0,28%	0,40%	0,53%	0,39%
T	-2,84 **	-0,90	-0,31	0,59	3,13 **	0,51
P	0,7%	19,4%	38,0%	56,7%	0,4%	61,8%
			Quota division			
ε	-1,55%	-0,78%	-0,56%	-0,12%	1,80%	-0,43%
S	0,82%	0,46%	0,34%	0,45%	0,69%	0,47%
T	-1,90 *	-1,71	-1,64	-0,26	2,62 *	-0,91
P	4,1%	5,6%	6,4%	79,8%	1,1%	38,0%
			Number division			
ε	-3,66%	-0,82%	-0,64%	-0,75%	2,43%	-0,64%
S	1,04%	0,75%	0,63%	0,45%	0,38%	0,43%
T	-3,51 **	-1,10	-1,02	-1,66	6,46 **	-1,51
P	0,2%	14,7%	16,5%	12,4%	0,0%	15,8%
			Size division			
ε	-1,31%	0,41%	0,14%	0,63%	1,02%	0,67%
S	0,48%	0,46%	0,57%	0,70%	0,66%	0,65%
T	-2,71 *	0,88	0,24	0,90	1,54	1,03
P	1,0%	19,8%	40,8%	38,7%	7,6%	32,5%

ε(1,3) Bias between group 1 and 3 for c = 4 based on 13 countries (12 for Size division)

	SD	HM	EP	MF	HA	LF
			Cluster division			
ε	-5,74%	-1,41%	-1,14%	-0,72%	3,51%	-0,82%
S	1,34%	0,90%	0,70%	0,51%	0,70%	0,40%
T	-4,29 **	-1,57	-1,63	-1,42	4,99 **	-2,07
P	0,1%	7,1%	6,5%	18,0%	0,0%	6,0%
			Quota division			
ε	-4,84%	-1,28%	-1,18%	-0,89%	3,44%	-1,21%
S	1,32%	0,76%	0,56%	0,27%	0,71%	0,34%
T	-3,68 **	-1,70	-2,10 *	-3,36 **	4,83 **	-3,60 **
P	0,2%	5,8%	2,9%	0,6%	0,0%	0,4%
			Number division			
ε	-6,43%	-1,12%	-0,89%	-0,01%	6,64%	-0,11%
S	1,55%	0,58%	0,53%	0,39%	1,42%	0,38%
T	-4,14 **	-1,95 *	-1,67	-0,02	4,67 **	-0,28
P	0,1%	3,8%	6,0%	98,8%	0,0%	78,1%
			Size division			
ε	-4,30%	-1,16%	-1,13%	-0,43%	3,05%	-0,51%
S	1,38%	0,92%	0,91%	0,54%	0,84%	0,50%
T	-3,11 **	-1,26	-1,24	-0,80	3,63 **	-1,02
P	0,5%	11,7%	12,0%	44,0%	0,2%	33,0%

Vector bias 4 groups

ε(1,4) Bias between group 1 and 4 for c = 4 based on 13 countries (12 for Size division)

	SD	HM	EP	MF	HA	LF
			Cluster division			
ε	-12,69%	-3,32%	-2,29%	0,97%	20,29%	1,24%
S	2,43%	1,43%	1,40%	0,99%	6,93%	1,07%
T	-5,22 **	-2,33 *	-1,63	0,98	2,93 **	1,15
P	0,0%	1,9%	6,4%	34,8%	0,6%	27,1%
			Quota division			
ε	-10,19%	-2,65%	-1,84%	0,86%	10,30%	0,97%
S	2,30%	1,10%	1,02%	0,47%	1,95%	0,32%
T	-4,43 **	-2,41 *	-1,80 *	1,82	5,28 **	3,04 *
P	0,0%	1,7%	4,8%	9,4%	0,0%	1,0%
			Number division			
ε	-21,56%	-11,36%	-6,55%	1,85%	18,82%	5,40%
S	6,17%	6,26%	3,21%	1,25%	4,05%	4,23%
T	-3,50 **	-1,81 *	-2,04 *	1,49	4,65 **	1,28
P	0,2%	4,7%	3,2%	16,3%	0,0%	22,6%
			Size division			
ε	-11,48%	-5,23%	-4,61%	-2,13%	12,57%	-1,73%
S	2,53%	2,23%	2,21%	2,32%	2,68%	2,50%
T	-4,54 **	-2,34 *	-2,08 *	-0,92	4,70 **	-0,69
P	0,0%	2,0%	3,1%	37,9%	0,0%	50,3%

ε(2,3) Bias between group 2 and 3 for c = 4 based on 13 countries (12 for Size division)

	SD	HM	EP	MF	HA	LF
			Cluster division			
ε	-3,91%	-0,98%	-1,05%	-0,98%	1,89%	-1,05%
S	0,87%	0,64%	0,62%	0,64%	0,47%	0,64%
T	-4,50 **	-1,52	-1,69	-1,53	3,99 **	-1,64
P	0,0%	7,7%	5,8%	15,3%	0,1%	12,7%
			Quota division			
ε	-3,21%	-0,50%	-0,62%	-0,80%	1,64%	-0,81%
S	0,63%	0,65%	0,60%	0,61%	0,63%	0,58%
T	-5,09 **	-0,78	-1,04	-1,32	2,62 *	-1,38
P	0,0%	22,5%	16,0%	21,1%	1,1%	19,2%
			Number division			
ε	-2,65%	-0,33%	-0,28%	0,72%	4,30%	0,52%
S	0,92%	0,57%	0,54%	0,45%	1,47%	0,50%
T	-2,88 **	-0,58	-0,51	1,61	2,92 **	1,02
P	0,7%	28,5%	30,8%	13,3%	0,6%	32,6%
			Size division			
ε	-2,93%	-1,57%	-1,28%	-1,13%	2,02%	-1,24%
S	1,14%	0,81%	0,82%	0,90%	0,94%	0,89%
T	-2,56 *	-1,94 *	-1,56	-1,24	2,14 *	-1,39
P	1,3%	3,9%	7,4%	23,9%	2,8%	19,3%

Vector bias 4 groups

ε(2,4) Bias between group 2 and 4 for c = 4 based on 13 countries (12 for Size division)

	SD	HM	EP	MF	HA	LF
			Cluster division			
ε	-10,70%	-2,90%	-2,21%	0,72%	18,97%	1,02%
S	1,97%	1,39%	1,45%	1,08%	7,00%	1,11%
T	-5,44 **	-2,09 *	-1,53	0,66	2,71 **	0,92
P	0,0%	2,9%	7,6%	52,1%	0,9%	37,6%
			Quota division			
ε	-8,40%	-1,86%	-1,29%	0,96%	8,73%	1,38%
S	1,58%	1,06%	1,05%	0,59%	1,59%	0,46%
T	-5,31 **	-1,76	-1,22	1,61	5,49 **	2,99 *
P	0,0%	5,2%	12,2%	13,4%	0,0%	1,1%
			Number division			
ε	-17,19%	-10,42%	-5,84%	2,56%	16,77%	6,03%
S	5,83%	6,10%	3,02%	1,30%	4,13%	4,19%
T	-2,95 **	-1,71	-1,93 *	1,97	4,06 **	1,44
P	0,6%	5,7%	3,9%	7,2%	0,1%	17,6%
			Size division			
ε	-10,05%	-5,73%	-4,84%	-2,89%	11,72%	-2,49%
S	2,53%	2,56%	2,58%	2,71%	2,54%	2,74%
T	-3,97 **	-2,24 *	-1,87 *	-1,07	4,62 **	-0,91
P	0,1%	2,3%	4,4%	30,9%	0,0%	38,3%

ε(3,4) Bias between group 3 and 4 for c = 4 based on 13 countries (12 for Size division)

	SD	HM	EP	MF	HA	LF
			Cluster division			
ε	-6,53%	-1,92%	-1,17%	1,62%	17,43%	2,00%
S	1,69%	1,31%	1,40%	1,28%	7,13%	1,31%
T	-3,86 **	-1,46	-0,83	1,27	2,44 *	1,52
P	0,1%	8,5%	21,1%	22,8%	1,5%	15,5%
			Quota division			
ε	-4,99%	-1,35%	-0,65%	1,73%	7,15%	2,15%
S	1,05%	0,77%	0,69%	0,51%	1,70%	0,45%
T	-4,73 **	-1,74	-0,94	3,39 **	4,22 **	4,78 **
P	0,0%	5,4%	18,3%	0,5%	0,1%	0,0%
			Number division			
ε	-14,27%	-9,95%	-5,51%	1,82%	13,07%	5,40%
S	5,79%	5,79%	2,82%	1,44%	3,94%	4,34%
T	-2,47 *	-1,72	-1,95 *	1,26	3,32 **	1,24
P	1,5%	5,6%	3,7%	23,0%	0,3%	23,7%
			Size division			
ε	-6,92%	-4,12%	-3,54%	-1,79%	9,82%	-1,25%
S	2,18%	2,49%	2,49%	2,68%	2,65%	2,57%
T	-3,18 **	-1,65	-1,42	-0,67	3,71 **	-0,49
P	0,4%	6,3%	9,1%	51,8%	0,2%	63,6%

Appendix 2:

Flow charts

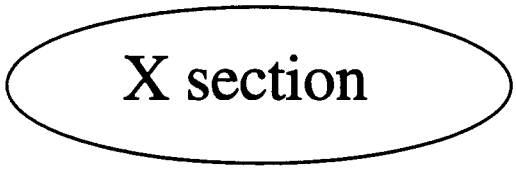
and

Pascal programs

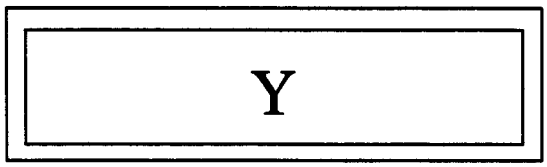
Explanation of flow chart symbols

= Start of program, section or procedure

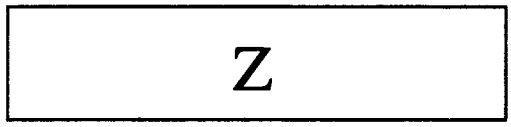
= End of program, section or procedure



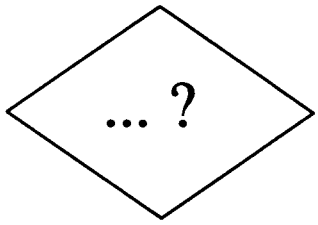
= Section with the name 'X'
(A section consists of several procedures)



= Procedure with the name 'Y'
which utilizes several other procedures



= Procedure with the name 'Z'



= Comparison made to determine further action

= Answer to comparison question

abcde... = Some places text is used to explain what is being done

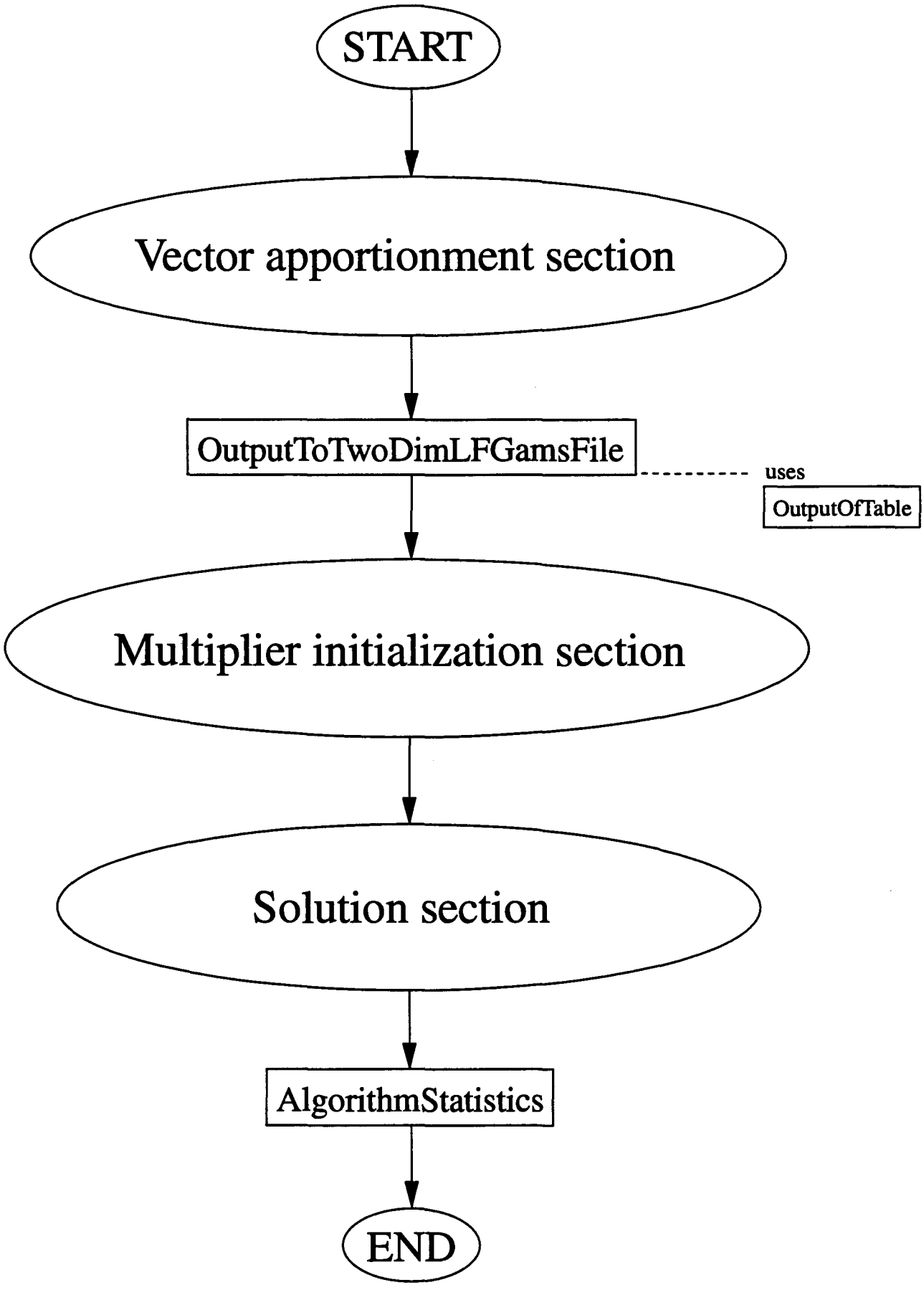
Example of an inputfile

The shown file is ICE63.DAT:

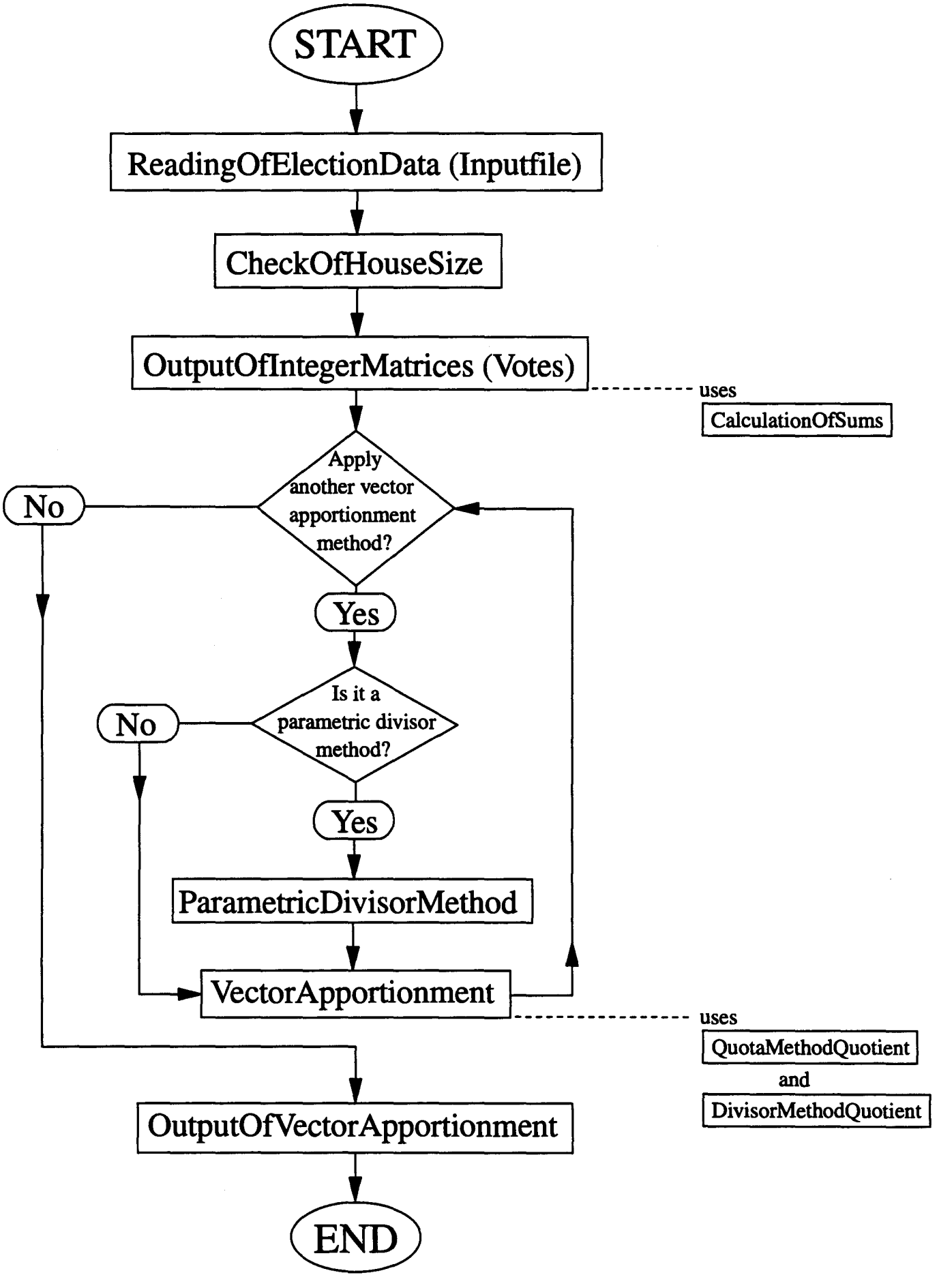
```
Rev 15
Ren  8
Vel  5
Vef  7
NoV  7
NoE  7
Aul  5
Sul  6

Alf  8
Fra 19
Sja 24
Alb  9

5730      6178     19122      6678
2804      2465      5040      1969
 912      2363      2019       739
 692      1743      1713       744
 537      2135      1765       663
1012      4530      2856      1621
 250      2804      1104       905
 760      2999      3402       955
```

The programs ElectionAlgorithm and MatrixBias are set up with ICE63.DAT as inputfile.

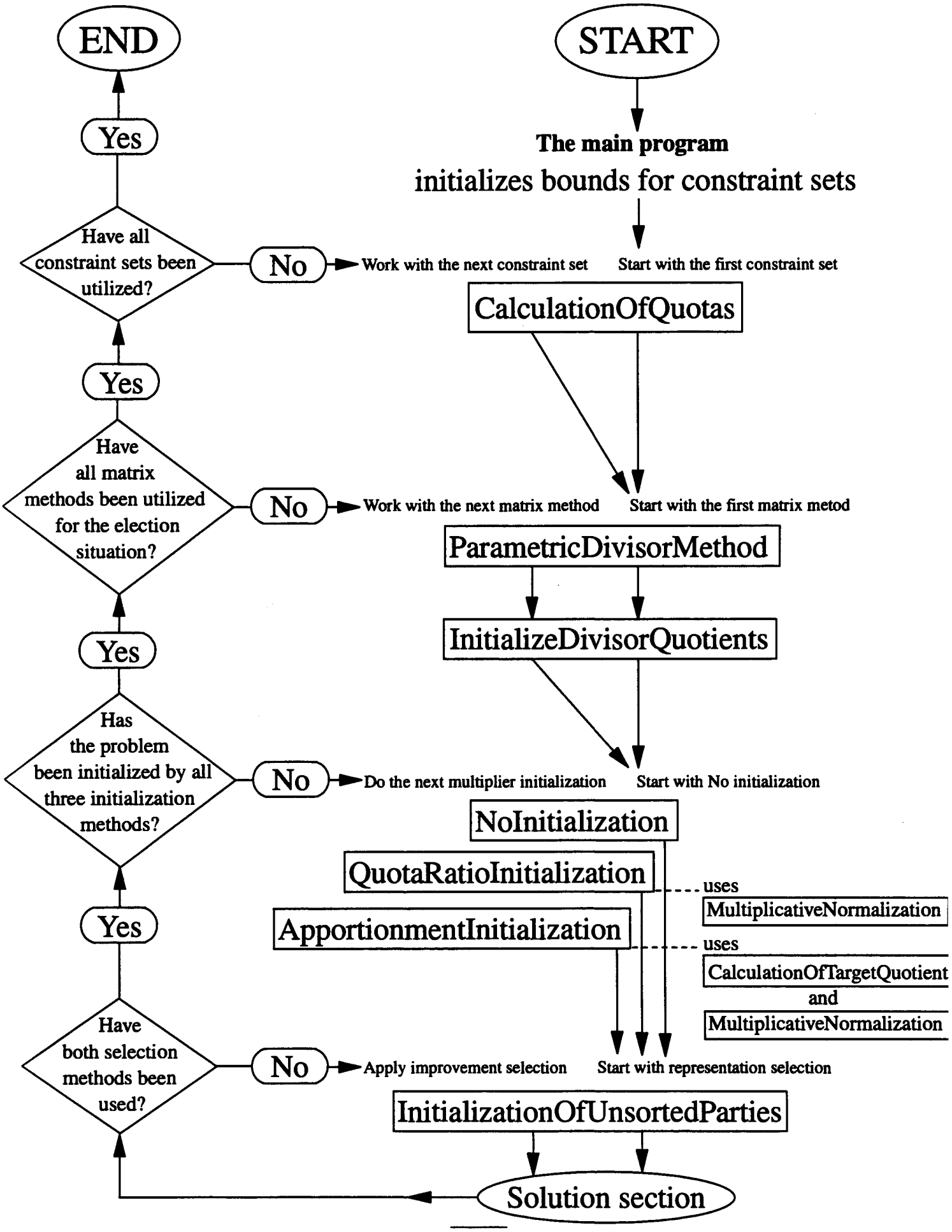
The ELECTION ALGORITHM program



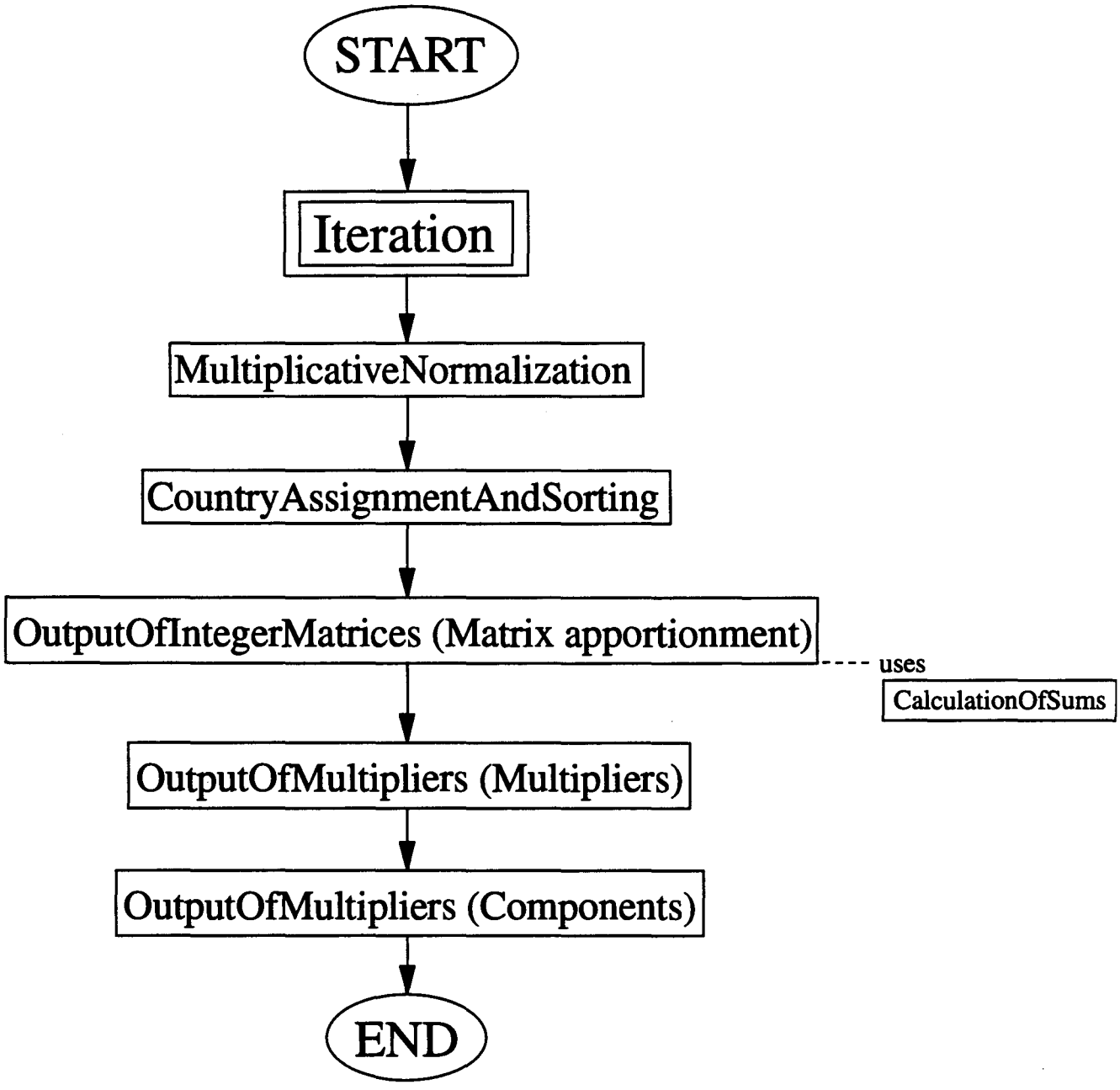
Vector apportionment section



Multiplier initialization section



Solution section

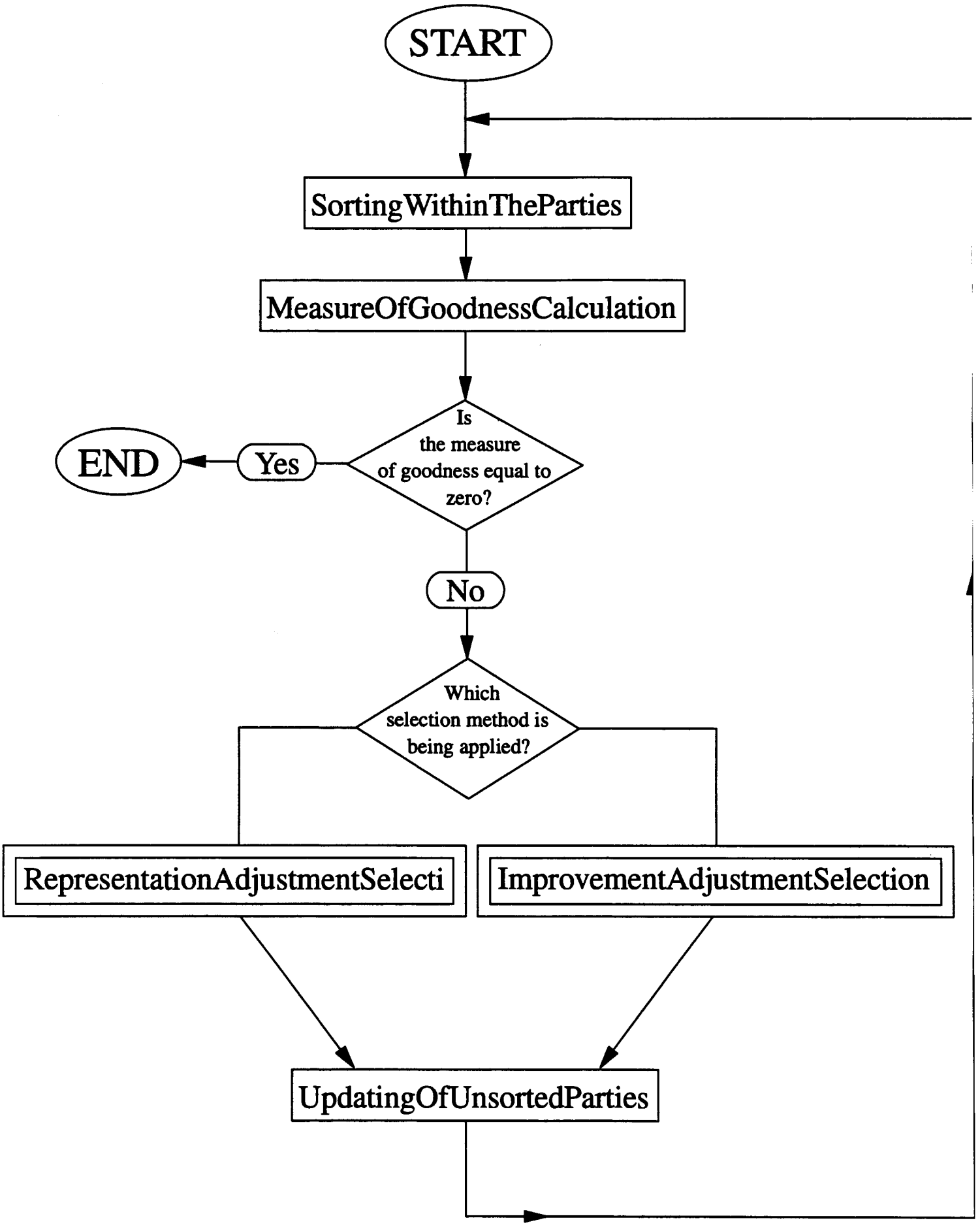


Comments regarding the flow charts on the next three pages:

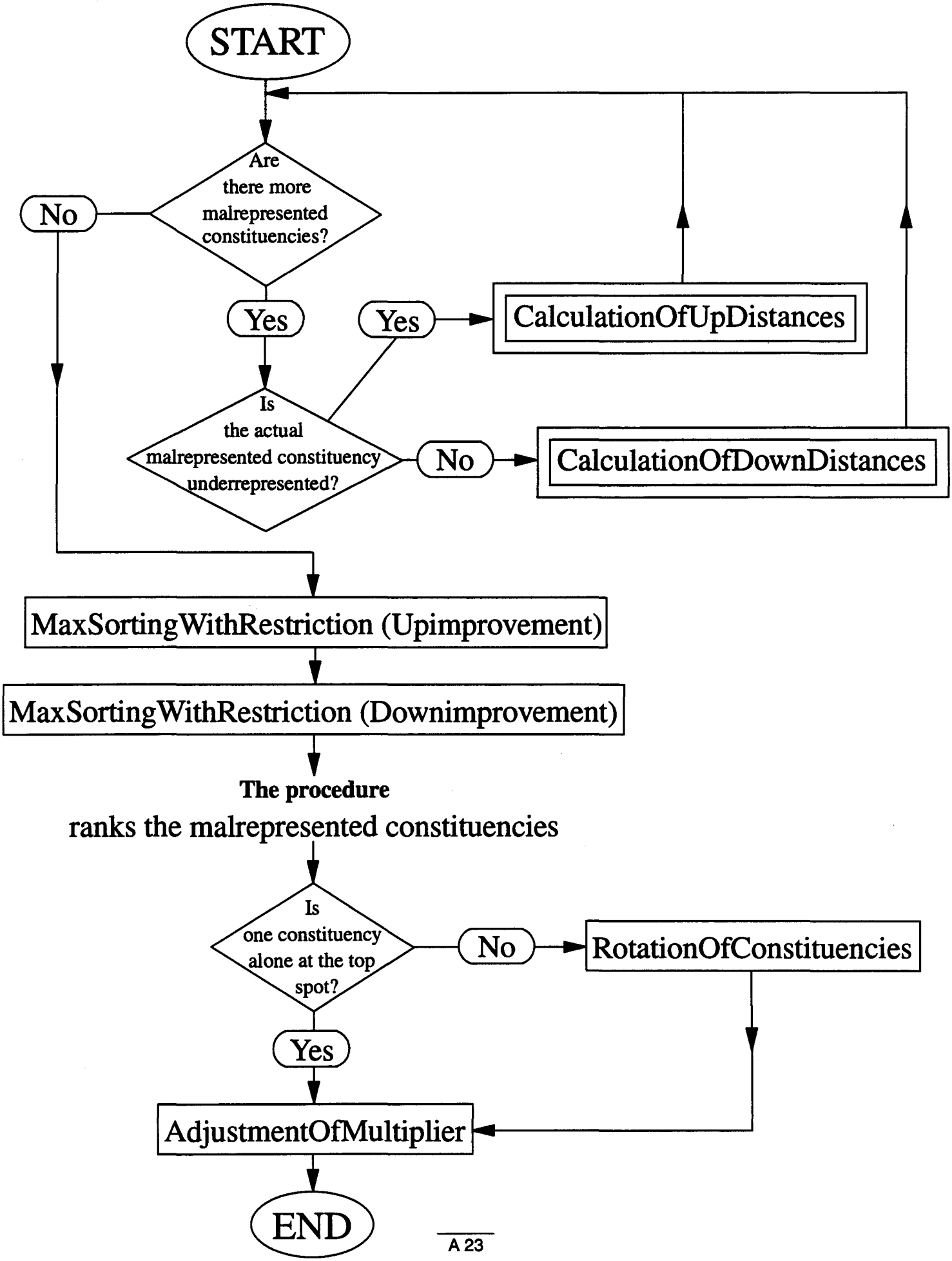
The Iteration procedure utilizes several procedures which themselves utilize several other procedures. Flow charts of the Iteration procedure and two of its most important subprocedures are presented on the following pages.

There is no flow chart of RepresentationAdjustmentSelecti since a flow chart for the more comprehensive procedure ImprovementAdjustmentSelection is presented. Furthermore we do not present a flow chart of CalculationOfDownDistances since this procedure is almost similar to CalculationOfUpDistances.

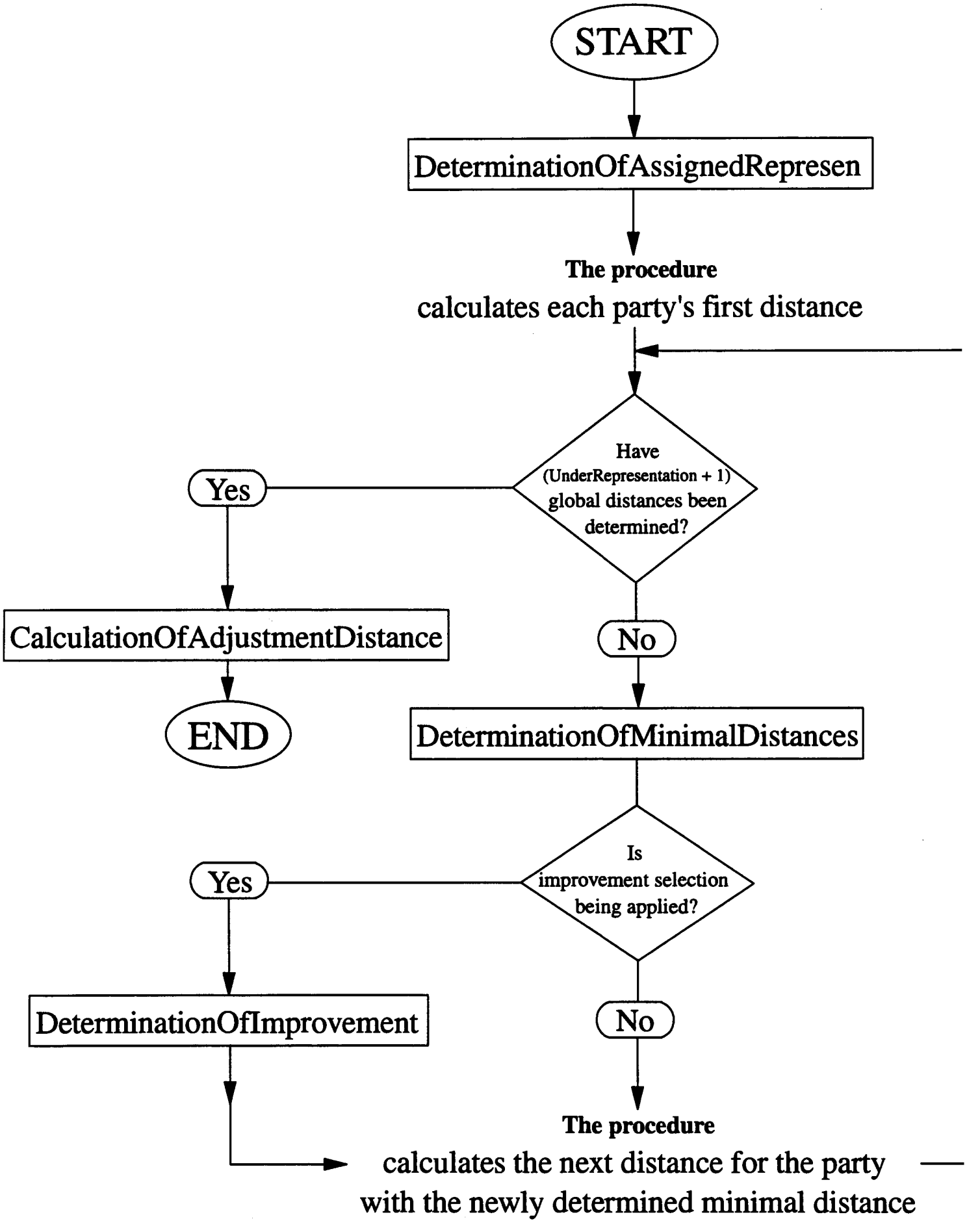
The Iteration procedure



The ImprovementAdjustmentSelection procedure



The CalculationOfUpDistances procedure



```
PROGRAM ElectionAlgorithm (Ice63, Result, BiasApport, TwoDimLFGams);

{The name of the inputfile (Ice63) has to be altered three places when the
 program is used for new data: 1. In the program declaration. 2. In the
 variable declaration. 3. In the call of the procedure ReadingOfData. The
 seek-replace procedure in EMACS is a simple way of doing this.}

{The letters i, j, ... are used as counting variables for loops.}

{The type double is used instead of real for real values to get better
 accuracy.}

{Constants, types and variables are declared sectionwise below.}

{Some of the procedures in the program are only used once.}

CONST NumberOfConstituencies =   8;
      NumberOfParties =          4;
      NumberOfDivisors =        30;
      NumberOfSeats =           60;
      WordLength =               3;
      NumberOfVectorMethods =    6;
      NumberOfConstraintSets =   3;
      NumberOfMatrixMethods =    4;
      NumberOfObjects =       (NumberOfConstituencies + NumberOfParties);

TYPE ConstituencyVector =     ARRAY [1..NumberOfConstituencies] OF integer;
     PartyVector =            ARRAY [1..NumberOfParties] OF integer;
     RealConstituencyVector = ARRAY [1..NumberOfConstituencies] OF double;
     RealPartyVector =        ARRAY [1..NumberOfParties] OF double;
     RealDivisorVector =      ARRAY [1..NumberOfDivisors] OF double;
     String =                 PACKED ARRAY [1..WordLength] OF char;
     Constituencies =         ARRAY [1..NumberOfConstituencies] OF String;
     Parties =                ARRAY [1..NumberOfParties] OF String;
     Methods =                ARRAY [1..NumberOfVectorMethods] OF String;
     IntegerMatrix =          ARRAY [1..NumberOfConstituencies,
                                     1..NumberOfParties] OF integer;
     MethodMatrix =           ARRAY [1..NumberOfObjects,
                                     1..NumberOfVectorMethods] OF integer;
     MultiplierMatrix =       ARRAY [1..NumberOfConstituencies, 1..3,
                                     1..NumberOfConstraintSets,
                                     1..NumberOfMatrixMethods] OF double;

VAR i:                          integer; {ConstituencyNumber}
    j:                          integer; {PartyNumber}
    k:                          integer; {RepresentativeNumber}
    l:                          integer; {RecordNumber}
    p:                          integer; {MatrixMethodNumber}
    c:                          integer; {ConstraintSetNumber}
    SumVotes:                   integer;
    HighestValue:               double;  {Used in sorting procedures.}
    Parameter:                  double;
    Result:                     text;
    ConstituencyVotes:          ConstituencyVector;
    ConstituencyRepresentatives: ConstituencyVector;
    PartyVotes:                 PartyVector;
    PartyRepresentatives:       PartyVector;
    ConstituencyMultiplier:     RealConstituencyVector;
    Divisor:                    RealDivisorVector;
    ConstituencyName:           Constituencies;
    PartyName:                  Parties;
    MethodName:                 Methods;
    Votes:                      IntegerMatrix;
    VectorRepresentatives:      MethodMatrix;
    Multiplier:                 MultiplierMatrix;
    Components:                 MultiplierMatrix;

{-------------------------------------------------------------------------------}

{Section for vector apportionment of constituencies and parties}

TYPE ObjectVector =     ARRAY [1..NumberOfObjects] OF integer;
     RealObjectVector = ARRAY [1..NumberOfObjects] OF double;
     Objects =          ARRAY [1..NumberOfObjects] OF String;

VAR m:             integer; {VectorMethodNumber}
    o:             integer; {ObjectNumber}
    Quota:         double;
    QuotaMethod:   boolean;
    DivisorMethod: boolean;
    Ice63:         text;
    ObjectVotes:   ObjectVector;
    ObjectName:    Objects;

{-------------------------------------------------------------------------------}
```

```
{Section for two dimensional LF}

VAR SumWholeSeats:                integer;
    QuotaViolation:               boolean;
    TwoDimLFGams:                 text;
    WholeConstituencySeats:       ConstituencyVector;
    FractionalConstituencySeats:  ConstituencyVector;
    WholePartySeats:              PartyVector;
    FractionalPartySeats:         PartyVector;
    IntegerParts:                 IntegerMatrix;
    Remainders:                   IntegerMatrix;
    RelevantCosts:                IntegerMatrix;
    WholeSeats:                   IntegerMatrix;

{-------------------------------------------------------------------------------}

{Section for initialization of multipliers}

CONST TargetWeight = 0.3;

TYPE ApportionmentInitializeMatrix = ARRAY [1..NumberOfConstraintSets,
                                            1..NumberOfMatrixMethods]
                                            OF double;
     NormalizationFactorMatrix =     ARRAY [1..3, 1..NumberOfConstraintSets,
                                            1..NumberOfMatrixMethods]
                                            OF double;

VAR n:                              integer; {InitializationNumber}
    f:                              integer; {NormalizationNumber}
    CountryQuota:                   double;
    InitialValueOfRepresentation:   double;
    ConstituencyQuota:              RealConstituencyVector;
    TargetQuotient:                 RealConstituencyVector;
    InitialMultiplier:              RealConstituencyVector;
    PartyQuota:                     RealPartyVector;
    MarginalValueOfRepresentation:  ApportionmentInitializeMatrix;
    NormalizationFactor:            NormalizationFactorMatrix;

{-------------------------------------------------------------------------------}

{Section for solving the matrix problem}

CONST AdjustmentWeight = 0.3;
      MaxIterations =    999;

TYPE ThreeDimensions =                ARRAY [1..NumberOfConstituencies,
                                             1..NumberOfParties,
                                             1..NumberOfDivisors] OF double;
     PartyRepresentativeData =        RECORD ConstituencyNumber: integer;
                                             Representative:     integer;
                                             Quotient:           double;
                                             END; {PartyRepresentativeData}
     PartyRecordVector =              ARRAY [1..NumberOfConstituencies,
                                             1..NumberOfParties,
                                             0..NumberOfDivisors]
                                             OF PartyRepresentativeData;
     SortedPartyRecordVector =        ARRAY [0..(NumberOfSeats + 1),
                                             1..NumberOfParties]
                                             OF PartyRepresentativeData;
     PartyData =                      RECORD ConstituencyNumber: integer;
                                             Magnitude:          double;
                                             END; {PartyData}
     AdjustmentData =                 RECORD Distance:           double;
                                             PartyNumber:        integer;
                                             ConstituencyNumber: integer;
                                             END; {AdjustmentData}
     PartyAndBenchmarkRecordVector = ARRAY [1..NumberOfDivisors] OF PartyData;
     AdjustmentRecordVector =        ARRAY [1..NumberOfDivisors]
                                             OF AdjustmentData;
```

```
      IterationMatrix =              ARRAY [1..2, 1..3,
                                           1..NumberOfConstraintSets,
                                           1..NumberOfMatrixMethods]
                                           OF integer;
      GoodnessMatrix =               ARRAY [1..3, 1..NumberOfConstraintSets,
                                           1..NumberOfMatrixMethods]
                                           OF integer;
      ApportionmentMatrix =          ARRAY [1..NumberOfConstituencies,
                                           1..NumberOfParties,
                                           1..NumberOfConstraintSets,
                                           1..NumberOfMatrixMethods]
                                           OF integer;
      RepresentativeData =           RECORD ConstituencyNumber: integer;
                                           PartyNumber:        integer;
                                           Representative:     integer;
                                           Quotient:           double;
                                           END; {RepresentativeData}
      CountryRecordVector =          ARRAY [1..NumberOfSeats]
                                           OF RepresentativeData;

VAR s:                               integer; {SelectionNumber}
    NumberOfIterations:              integer;
    MeasureOfGoodness:               integer;
    InitialMeasureOfGoodness:        integer;
    ActualConstituencyNumber:        integer;
    DistanceConstituencyNumber:      integer;
    DistancePartyNumber:             integer;
    UpRotationConstituencyNumber:    integer;
    DownRotationConstituencyNumber:  integer;
    GlobalDistanceNumber:            integer;
    SelectedConstituencyNumber:      integer;
    MaxUnderRepresentation:          integer;
    MaxOverRepresentation:           integer;
    UnderReprConstituencyNumber:     integer;
    OverReprConstituencyNumber:      integer;
    MultiplierNumber:                integer;
    NotFinished:                     boolean;
    NotAbandoned:                    boolean;
    SeveralWithMaxUnderRepr:         boolean;
    SeveralWithMaxOverRepr:          boolean;
    UpAdjustmentLastIteration:       boolean;
    BiasApport:                      text;
    AssignedRepresentatives:         ConstituencyVector;
    OverRepresentation:              ConstituencyVector;
    UnderRepresentation:             ConstituencyVector;
    RemainderMalRepresentation:      ConstituencyVector;
    UpImprovement:                   ConstituencyVector;
    DownImprovement:                 ConstituencyVector;
    PositionWithinParty:             PartyVector;
    AssignedConstituencyRepr:        PartyVector;
    DistanceNumber:                  PartyVector;
    Increase:                        RealConstituencyVector;
    Decrease:                        RealConstituencyVector;
    DivisorQuotients:                ThreeDimensions;
    UnsortedParties:                 PartyRecordVector;
    SortedParties:                   SortedPartyRecordVector;
    Distance:                        PartyAndBenchmarkRecordVector;
    Adjustment:                      AdjustmentRecordVector;
    Representatives:                 IntegerMatrix;
    Iterations:                      IterationMatrix;
    Failures:                        IterationMatrix;
    Goodness:                        GoodnessMatrix;
    Apportionment:                   ApportionmentMatrix;
    SortedCountry:                   CountryRecordVector;

{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}
```

```
PROCEDURE ReadingOfElectionData (VAR ElectionData: text);

{The way of reading below requires that the data in the inputfile are ordered
 in a specific way. The data for each party and for each constituency have
 their own line. The election apportionment data must not appear before in
 position WordLength + 1 of the inputfile. To make the inputfile more readable
 it has one blank line between the last party data and the first constituency
 data and between the last constituency data and the vote matrix.}

BEGIN {ReadingOfElectionData}

reset (ElectionData);

FOR i:= 1 TO NumberOfConstituencies DO
   readln (ElectionData, ConstituencyName [i], VectorRepresentatives [i,1]);
readln (ElectionData);

FOR j:= 1 TO NumberOfParties DO
   readln (ElectionData, PartyName [j],
      VectorRepresentatives [(NumberOfConstituencies + j), 1]);
readln (ElectionData);

FOR i:= 1 TO NumberOfConstituencies DO BEGIN
   FOR j:= 1 TO NumberOfParties DO
      read (ElectionData, Votes [i,j]);
   readln (ElectionData);
END; {For i}

END; {ReadingOfElectionData}

{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}

PROCEDURE CheckOfHouseSize (VAR OutputFile: text);

{Checks that the given NumberOfSeats is consistent with the supplied election
 day apportionment. A bug is entered into the program to stop its execution if
 the value of NumberOfSeats is incorrect.}

VAR CheckSum: integer;

BEGIN {CheckOfHouseSize}

CheckSum:= 0;
FOR i:= 1 TO NumberOfConstituencies DO
   CheckSum:= CheckSum + VectorRepresentatives [i,1];

IF CheckSum <> NumberOfSeats THEN BEGIN
   writeln (OutputFile, 'NumberOfSeats has got a wrong value ');
   writeln (OutputFile);
   MethodName [NumberOfVectorMethods + 1] := 'Bug' ;
END; {Then}

CheckSum:= 0;
FOR j:= 1 TO NumberOfParties DO
   CheckSum:= CheckSum +
      VectorRepresentatives [(j + NumberOfConstituencies), 1];

IF CheckSum <> NumberOfSeats THEN BEGIN
   writeln (OutputFile, 'NumberOfSeats has got a wrong value ');
   writeln (OutputFile);
   MethodName [NumberOfVectorMethods + 1] := 'Bug' ;
END; {Then}

END; {CheckOfHouseSize}

{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}

PROCEDURE WritingOfLine (VAR OutputFile: text);

BEGIN {WritingOfLine}

writeln (OutputFile, '-------------------------------------------------------',
   '------------------------');

END; {WritingOfLine}

{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}
```

```
PROCEDURE CalculationOfSums (Matrix:              IntegerMatrix;
                             VAR ConstituencySum: ConstituencyVector;
                             VAR PartySum:        PartyVector;
                             VAR Sum:             integer);

BEGIN {CalculationOfSums}

FOR i:= 1 TO NumberOfConstituencies DO
   ConstituencySum [i] := 0;

FOR j:= 1 To NumberOfParties DO
   PartySum [j] := 0;

Sum:= 0;

FOR i:= 1 TO NumberOfConstituencies DO
   FOR j:= 1 TO NumberOfParties DO
      ConstituencySum [i] := ConstituencySum [i] + Matrix [i,j];

FOR j:= 1 TO NumberOfParties DO
   FOR i:= 1 TO NumberOfConstituencies DO
      PartySum [j] := PartySum [j] + Matrix [i,j];

FOR i:= 1 TO NumberOfConstituencies DO
   FOR j:= 1 TO NumberOfParties DO
      Sum:= Sum + Matrix [i,j];

END; {CalculationOfSums}

{-------------------------------------------------------------------------------}
{-------------------------------------------------------------------------------}

PROCEDURE OutputOfIntegerMatrices (VAR OutputFile: text;
                                   FieldWidth:     integer;
                                   SumFieldWidth:  integer;
                                   Matrix:         IntegerMatrix);

VAR s:                integer;
    t:                integer;
    RemainingParties: integer;
    ColumnsPerLine:   integer;
    NumberOfTables:   integer;
    Sum:              integer;
    ConstituencySum:  ConstituencyVector;
    PartySum:         PartyVector;

BEGIN {OutputOfIntegerMatrices}

CalculationOfSums (Matrix, ConstituencySum, PartySum, Sum);

RemainingParties:= NumberOfParties;
ColumnsPerLine:= ((77 - SumFieldWidth) DIV FieldWidth) + 1;
IF (NumberOfParties + 1) MOD ColumnsPerLine = 0 THEN
   NumberOfTables:= ((NumberOfParties + 1) DIV ColumnsPerLine)
ELSE NumberOfTables:= ((NumberOfParties + 1) DIV ColumnsPerLine) + 1;

writeln (OutputFile);
FOR t:= 1 TO NumberOfTables DO BEGIN
   write (OutputFile, '   ');
   FOR j:= (NumberOfParties - RemainingParties + 1) TO MIN(NumberOfParties,
            (NumberOfParties - RemainingParties + ColumnsPerLine)) DO
      write (OutputFile, PartyName [j] :FieldWidth);
   IF t = NumberOfTables THEN BEGIN
      FOR s:= 1 TO (SumFieldWidth - 3) DO
         write (OutputFile, ' ');
      writeln (OutputFile, 'Sum');
   END {Then}
   ELSE writeln (OutputFile);
```

```
   FOR i:= 1 TO NumberOfConstituencies DO BEGIN
      write (OutputFile, ConstituencyName [i]);
      FOR j:= (NumberOfParties - RemainingParties + 1) TO MIN(NumberOfParties,
               (NumberOfParties - RemainingParties + ColumnsPerLine)) DO
         write (OutputFile, Matrix [i,j] :FieldWidth);
      IF t = NumberOfTables THEN
         writeln (OutputFile, ConstituencySum [i] :SumFieldWidth)
      ELSE writeln (OutputFile);
   END; {For i}
   writeln (OutputFile);

   write (OutputFile, 'Sum');
   FOR j:= (NumberOfParties - RemainingParties + 1) TO MIN(NumberOfParties,
           (NumberOfParties - RemainingParties + ColumnsPerLine)) DO
      write (OutputFile, PartySum [j] :FieldWidth);
   IF t = NumberOfTables THEN
      writeln (OutputFile, Sum:SumFieldWidth)
   ELSE BEGIN
      writeln (OutputFile);
      writeln (OutputFile);
   END; {Else}
   RemainingParties:= RemainingParties - ColumnsPerLine;
   writeln (OutputFile);
END; {For t}

END; {OutputOfIntegerMatrices}

{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}

PROCEDURE ParametricDivisorMethod;

{The procedure initiates the divisors for the parametric divisor method. The
 parameter is a real number between 0 and 1.}

BEGIN {ParametricDivisorMethod}

FOR k:= 1 TO NumberOfDivisors DO
   Divisor [k]:= k - 1 + Parameter;

END; {ParametricDivisorMethod}

{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}

PROCEDURE QuotaMethodQuotient (CellVotes:      integer;
                               QuotientNumber: integer;
                               VAR Quotient:   double);

{Another possibility is to include the variable Quota in the procedure call}

BEGIN {QuotaMethodQuotient}

Quotient:= CellVotes - (QuotientNumber - 1) * Quota;

END; {QuotaMethodQuotient}

{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}

PROCEDURE DivisorMethodQuotient (CellVotes:      integer;
                                 QuotientNumber: integer;
                                 VAR Quotient:   double);

BEGIN {DivisorMethodQuotient}

IF CellVotes > 0 THEN
   IF Divisor [QuotientNumber] > 0 THEN
      Quotient:= LN(CellVotes / Divisor [QuotientNumber])
   ELSE Quotient:= 50 / QuotientNumber
ELSE Quotient:= -50;

END; {DivisorMethodQuotient}

{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}
```

```
PROCEDURE VectorApportionment (FirstObject: integer;
                               LastObject:  integer);

VAR BestObjectNumber: integer;
    ObjectQuotient:   RealObjectVector;

BEGIN {VectorApportionment}

FOR o:= FirstObject TO LastObject DO BEGIN
   VectorRepresentatives [o,m] := 0;
   IF QuotaMethod THEN
      QuotaMethodQuotient (ObjectVotes [o], 1, ObjectQuotient [o])
   ELSE IF DivisorMethod THEN
           DivisorMethodQuotient (ObjectVotes [o], 1, ObjectQuotient [o]);
END; {For o}

l:= 1;
WHILE l <= NumberOfSeats DO BEGIN
   HighestValue:= -1.0E+20;
   FOR o:= FirstObject TO LastObject DO
      IF ObjectQuotient [o] > HighestValue THEN BEGIN
         HighestValue:= ObjectQuotient [o];
         BestObjectNumber:= o;
      END; {Then}

   VectorRepresentatives [BestObjectNumber, m] :=
      VectorRepresentatives [BestObjectNumber, m] + 1;

   IF DivisorMethod THEN
      DivisorMethodQuotient (ObjectVotes [BestObjectNumber],
        (VectorRepresentatives [BestObjectNumber, m] + 1),
         ObjectQuotient [BestObjectNumber])
   ELSE IF QuotaMethod THEN
           QuotaMethodQuotient (ObjectVotes [BestObjectNumber],
             (VectorRepresentatives [BestObjectNumber, m] + 1),
              ObjectQuotient [BestObjectNumber]);

   l:= l + 1;
END; {While}

END; {VectorApportionment}

{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}

PROCEDURE OutputOfVectorApportionment (VAR OutputFile: text;
                                       FirstObject:    integer;
                                       LastObject:     integer);

BEGIN {OutputOfVectorApportionment}

writeln (OutputFile);
write   (OutputFile, '     ');
FOR m:= 1 TO NumberOfVectorMethods DO
   write (OutputFile, '  ', MethodName [m]);
writeln (OutputFile);

FOR o:= FirstObject TO LastObject DO BEGIN
   write (OutputFile, ObjectName [o], ' ');
   FOR m:= 1 TO NumberOfVectorMethods DO
      write (OutputFile, VectorRepresentatives [o,m]:4, ' ');
   writeln (OutputFile);
END; {For o}
writeln (OutputFile);

END; {OutputOfVectorApportionment}

{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}
```

```
PROCEDURE OutputOfTable (VAR GamsFile: text;
                         FirstParty:   integer;
                         LastParty:    integer);

BEGIN {OutputOfTable}

FOR j:= FirstParty TO LastParty DO
   write (GamsFile, PartyName [j] :10);
writeln (GamsFile);

FOR i:= 1 TO NumberOfConstituencies DO BEGIN
   write (GamsFile, ConstituencyName [i]);
   FOR j:= FirstParty TO LastParty DO
      write (GamsFile, RelevantCosts [i,j] :10);
   IF (i = NumberOfConstituencies) AND (j = NumberOfParties) THEN
      writeln (GamsFile, ' ;')
   ELSE writeln (GamsFile);
END; {For i}

END; {OutputOfTable}

{-------------------------------------------------------------------------------}
{-------------------------------------------------------------------------------}

PROCEDURE OutputToTwoDimLFGamsFile (VAR GamsFile: text);

{Creates a gams file for solving the controlled rounding of the internal
 entries in the initial fair share matrix, i.e. two dimensional LF}

BEGIN {OutputToTwoDimLFGamsFile}

writeln (GamsFile, 'SETS');
writeln (GamsFile);
writeln (GamsFile, '   I constituencies /');
writeln (GamsFile);
FOR i:= 1 TO NumberOfConstituencies DO BEGIN
   write (GamsFile, ConstituencyName [i]);
   IF (i < NumberOfConstituencies) THEN
      writeln (GamsFile)
   ELSE writeln (GamsFile, ' /');
END; {For i}
writeln (GamsFile);
writeln (GamsFile, '   J parties /');
writeln (GamsFile);
FOR j:= 1 TO NumberOfParties DO BEGIN
   write (GamsFile, PartyName [j]);
   IF (j < NumberOfParties) THEN
      writeln (GamsFile)
   ELSE writeln (GamsFile, ' / ;');
END; {For j}
writeln (GamsFile);
writeln (GamsFile);

writeln (GamsFile, 'PARAMETERS');
writeln (GamsFile);
writeln (GamsFile, '   C(I) number of seats for constituency i /');
writeln (GamsFile);
FOR i:= 1 TO NumberOfConstituencies DO BEGIN
   write (GamsFile, ConstituencyName [i],
      FractionalConstituencySeats [i] :3);
   IF (i < NumberOfConstituencies) THEN
      writeln (GamsFile)
   ELSE writeln (GamsFile, ' /');
END; {For i}
writeln (GamsFile);
writeln (GamsFile, '   P(J) number of seats for party j /');
writeln (GamsFile);
```

```
FOR j:= 1 TO NumberOfParties DO BEGIN
   write (GamsFile, PartyName [j], FractionalPartySeats [j] :3);
   IF j < NumberOfParties THEN
      writeln (GamsFile)
   ELSE writeln (GamsFile, ' / ;');
END; {For j}
writeln (GamsFile);
writeln (GamsFile);

writeln (GamsFile, 'TABLE R(I,J) relevant costs for the cells ');
writeln (GamsFile);
write (GamsFile, '    ');
OutputOfTable (GamsFile, 1, MIN(6, NumberOfParties));
writeln (GamsFile);
writeln (GamsFile);

IF NumberOfParties > 6 THEN BEGIN
   write (GamsFile, ' + ');
   OutputOfTable (GamsFile, 7, NumberOfParties);
   writeln (GamsFile);
   writeln (GamsFile);
END; {Then}

{The sequence above must be repeated if there are more than 12 parties}

writeln (GamsFile, 'VARIABLES');
writeln (GamsFile);
writeln (GamsFile, '   A(I,J)  seat for party j in constituency i');
writeln (GamsFile, '   Z       object value ;');
writeln (GamsFile);
writeln (GamsFile);

writeln (GamsFile, 'POSITIVE VARIABLE A ;');
writeln (GamsFile);
writeln (GamsFile);

writeln (GamsFile, 'EQUATIONS');
writeln (GamsFile);
writeln (GamsFile, 'VALUE   defines the objective function');
writeln (GamsFile, 'CONSTIT seats for the constituencies');
writeln (GamsFile, 'PARTY   seats for the parties');
writeln (GamsFile, 'SEAT    a maximum of one seat per cell ;');
writeln (GamsFile);
writeln (GamsFile, 'VALUE..      Z    =E= SUM( (I,J), R(I,J) * A(I,J) );');
writeln (GamsFile, 'CONSTIT(I).. C(I) =E= SUM( J, A(I,J) );');
writeln (GamsFile, 'PARTY(J)..   P(J) =E= SUM( I, A(I,J) );');
writeln (GamsFile, 'SEAT(I,J)..  A(I,J) =L= 1 ;');
writeln (GamsFile);
writeln (GamsFile);

writeln (GamsFile, 'MODEL TWODIMLF / ALL / ;');
writeln (GamsFile);
writeln (GamsFile, 'OPTION LIMROW = 0 ;');
writeln (GamsFile, 'OPTION LIMCOL = 0 ;');
writeln (GamsFile);
writeln (GamsFile, 'OPTION ITERLIM = 2000 ;');
writeln (GamsFile);
writeln (GamsFile);
writeln (GamsFile, 'SOLVE TWODIMLF USING LP MINIMIZING Z ;');

END; {OutputToTwoDimLFGamsFile}

{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}
```

```
PROCEDURE CalculationOfQuotas;

BEGIN {CalculationOfQuotas}

CountryQuota:= SumVotes / NumberOfSeats;

FOR i:= 1 TO NumberOfConstituencies DO
   IF ConstituencyRepresentatives [i] > 0 THEN
      ConstituencyQuota [i] :=
         ConstituencyVotes [i] / ConstituencyRepresentatives [i]
   ELSE ConstituencyQuota [i] := 1.0E+5;

FOR j:= 1 TO NumberOfParties DO
   IF PartyRepresentatives [j] > 0 THEN
      PartyQuota [j] := PartyVotes [j] / PartyRepresentatives [j]
   ELSE PartyQuota [j] := 1.0E+5;

END; {CalculationOfQuotas}

{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}

PROCEDURE MultiplicativeNormalization (VAR NormalizationFactor: double);

VAR MultiplierProduct: double;

BEGIN {MultiplicativeNormalization}

MultiplierProduct:= 1;
FOR i:= 1 TO NumberOfConstituencies DO
   MultiplierProduct:= MultiplierProduct * ConstituencyMultiplier [i];

NormalizationFactor:=
   EXP( (1 / NumberOfConstituencies) * LN(1 / MultiplierProduct));

END; {MultiplicativeNormalization}

{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}

PROCEDURE NoInitialization;

BEGIN {NoInitialization}

{This initialization lets all ConstituencyMultipliers be equal.}

FOR i:= 1 TO NumberOfConstituencies DO
   ConstituencyMultiplier [i] := 1;

END; {NoInitialization}

{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}

PROCEDURE QuotaRatioInitialization;

BEGIN {QuotaRatioInitialization}

{The initialization makes the persons per seat ratio equal for all
 constituencies.}

FOR i:= 1 TO NumberOfConstituencies DO
   ConstituencyMultiplier [i]:= CountryQuota / ConstituencyQuota [i];

MultiplicativeNormalization (NormalizationFactor [1,c,p]);

FOR i:= 1 TO NumberOfConstituencies DO
   Multiplier [i,1,c,p]:=
      ConstituencyMultiplier [i]  * NormalizationFactor [1,c,p];

END; {QuotaRatioInitialization}

{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}
```

```
PROCEDURE CalculationOfTargetQuotients;

{Sorts the (ConstituencyRepresentatives [i] + 1) largest quotients within each
 constituency and calculates the TargetQuotients.}

VAR NumberBestParty:                  integer;
    LastConstituencyQuotientWith:     double;
    BestConstituencyQuotientWithout: double;
    ActiveRepresentative:             PartyVector;
    PartyQuotient:                    RealPartyVector;

BEGIN {CalculationOfTargetQuotients}

FOR i:= 1 TO NumberOfConstituencies DO BEGIN

   l:= 1;
   FOR j:= 1 TO NumberOfParties DO BEGIN
      ActiveRepresentative [j] := 1;
      DivisorMethodQuotient (Votes [i,j], ActiveRepresentative [j],
                             PartyQuotient [j]);
   END; {For j}

   WHILE l <= (ConstituencyRepresentatives [i] + 1) DO BEGIN
      HighestValue:= -1.0E+20;
      FOR j:= 1 TO NumberOfParties DO
         IF PartyQuotient [j] > HighestValue THEN BEGIN
            HighestValue:= PartyQuotient [j];
            NumberBestParty:= j;
         END; {Then}

      ActiveRepresentative [NumberBestParty] :=
         ActiveRepresentative [NumberBestParty] + 1;
      DivisorMethodQuotient (Votes [i, NumberBestParty],
                             ActiveRepresentative [NumberBestParty],
                             PartyQuotient [NumberBestParty]);

      IF l = ConstituencyRepresentatives [i] THEN
         LastConstituencyQuotientWith:= HighestValue;
      IF l = (ConstituencyRepresentatives [i] + 1) THEN
         BestConstituencyQuotientWithout:= HighestValue;

      l:= l + 1;
   END; {While}

   TargetQuotient [i] := TargetWeight * EXP(LastConstituencyQuotientWith) +
                    (1 - TargetWeight) * EXP(BestConstituencyQuotientWithout);
END; {For i}

END; {CalculationOfTargetQuotients}

{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}

PROCEDURE ApportionmentInitialization;

BEGIN {ApportionmentInitialization}

{The initialization is based on the vector apportionment within each
 constituency. The value of InitialValueOfRepresentation may be chosen
 arbitrarily.}

InitialValueOfRepresentation:= SumVotes /
  (NumberOfSeats + (1 - TargetWeight) * NumberOfConstituencies);
CalculationOfTargetQuotients;

FOR i:= 1 TO NumberOfConstituencies DO
   ConstituencyMultiplier [i]:=
      InitialValueOfRepresentation / TargetQuotient [i];

MultiplicativeNormalization (NormalizationFactor [2,c,p]);

{The other alternative is AdditiveNormalization}

FOR i:= 1 TO NumberOfConstituencies DO BEGIN
   ConstituencyMultiplier [i] :=
      ConstituencyMultiplier [i] * NormalizationFactor [2,c,p];
   Multiplier [i,2,c,p]:= ConstituencyMultiplier [i];
END; {For i}

MarginalValueOfRepresentation [c,p] :=
   LN(InitialValueOfRepresentation) + LN(NormalizationFactor [2,c,p]);

END; {ApportionmentInitialization}

{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}
```

```
PROCEDURE InitializeDivisorQuotients;

BEGIN {InitializeDivisorQuotients}

FOR i:= 1 TO NumberOfConstituencies DO
   FOR j:= 1 TO NumberOfParties DO
      FOR k:= 1 TO NumberOfDivisors DO
         DivisorMethodQuotient (Votes [i,j], k, DivisorQuotients [i,j,k]);

END; {InitializeDivisorQuotients}

{-------------------------------------------------------------------------------}
{-------------------------------------------------------------------------------}

PROCEDURE InitializationOfUnsortedParties;

BEGIN {InitializationOfUnsortedParties}

FOR i:= 1 TO NumberOfConstituencies DO
   FOR j:= 1 TO NumberOfParties DO BEGIN
      UnsortedParties [i,j,0].Quotient:= -100;
      FOR k:= 1 TO NumberOfDivisors DO BEGIN
         UnsortedParties [i,j,k].Quotient:=
            LN(ConstituencyMultiplier [i]) + DivisorQuotients [i,j,k];
         UnsortedParties [i,j,k].ConstituencyNumber:= i;
         UnsortedParties [i,j,k].Representative:= k;
      END; {For k}
   END; {For j}

END; {InitializationOfUnsortedParties}

{-------------------------------------------------------------------------------}
{-------------------------------------------------------------------------------}

PROCEDURE SortingWithinTheParties;

{Record number 0 of UnsortedParties is a helprecord with value -100. There are
 a total of NumberOfDivisors * NumberOfConstituencies quotients to sort for
 each party. However, we only sort the (NumberOfSeats + 1) largest. This "stop
 value" is usually high enough. However, if a constituency is big compared to
 the other constituencies (e.g. shall have more than 50% of the seats), it will
 dominate this sorted party list. This may cause problems in the procedure
 CalculationOfDownDistances because some actual IncomingPartyRepresentatives
 may not be among the (NumberOfSeats + 1) largest quotients, with the result
 that the program crashes.}

VAR BestConstituencyNumber: integer;
    ActiveRepresentative:   ConstituencyVector;
    BestRecord:             PartyRepresentativeData;

BEGIN {SortingWithinTheParties}

FOR i:= 1 TO NumberOfConstituencies DO
   AssignedRepresentatives [i] := 0;

FOR j:= 1 TO NumberOfParties DO BEGIN
   l:= 1;
   FOR i:= 1 TO NumberOfConstituencies DO
      ActiveRepresentative [i] := 1 ;

   WHILE l <= NumberOfSeats DO BEGIN
      HighestValue:= -100;
      FOR i:= 1 TO NumberOfConstituencies DO
         IF UnsortedParties [i, j, ActiveRepresentative [i]].Quotient >
            HighestValue THEN BEGIN
            HighestValue:=
               UnsortedParties [i, j, ActiveRepresentative [i]].Quotient;
            BestRecord:= UnsortedParties [i, j, ActiveRepresentative [i]];
            BestConstituencyNumber:= i;
         END; {Then}

      IF ActiveRepresentative [BestConstituencyNumber] < NumberOfDivisors THEN
         ActiveRepresentative [BestConstituencyNumber] :=
            ActiveRepresentative [BestConstituencyNumber] + 1
      ELSE ActiveRepresentative [BestConstituencyNumber] := 0;

      IF l = PartyRepresentatives [j] THEN
         FOR i:= 1 TO NumberOfConstituencies DO
            AssignedRepresentatives [i] :=
               AssignedRepresentatives [i] + (ActiveRepresentative [i] - 1);

      SortedParties [l,j] := BestRecord;
      l:= l + 1;
   END; {While l}

END; {For j}

END; {SortingWithinTheParties}

{-------------------------------------------------------------------------------}
{-------------------------------------------------------------------------------}
```

```
PROCEDURE MeasureOfGoodnessCalculation;

BEGIN {MeasureOfGoodnessCalculation}

MeasureOfGoodness:= 0;
FOR i:= 1 TO NumberOfConstituencies DO BEGIN
   UnderRepresentation [i] := 0;
   OverRepresentation [i] := 0;
END; {For i}

FOR i:= 1 TO NumberOfConstituencies DO BEGIN
   IF AssignedRepresentatives [i] < ConstituencyRepresentatives [i] THEN
      UnderRepresentation [i] :=
         ConstituencyRepresentatives [i] - AssignedRepresentatives [i]
   ELSE IF AssignedRepresentatives [i] > ConstituencyRepresentatives [i] THEN
           OverRepresentation [i] :=
              AssignedRepresentatives [i] - ConstituencyRepresentatives [i];
   MeasureOfGoodness:=
      MeasureOfGoodness + UnderRepresentation [i] + OverRepresentation [i];
END; {For i}

END; {MeasureOfGoodnessCalculation}

{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}

PROCEDURE DeterminationOfAssignedRepresen (ConstituencyNumber: integer);

BEGIN {DeterminationOfAssignedRepresentatives}

l:= PartyRepresentatives [j];
WHILE (SortedParties [l,j].ConstituencyNumber <> ConstituencyNumber)
   AND (l > 0) DO
   l:= l - 1;

IF l = 0 THEN
   AssignedConstituencyRepr [j] := 0
ELSE AssignedConstituencyRepr [j] := SortedParties [l,j].Representative;

END; {DeterminationOfAssignedRepresentatives}

{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}

PROCEDURE DeterminationOfMinimalDistances;

VAR MinimalDistance: double;

BEGIN {DeterminationOfMinimalDistances}

MinimalDistance:= 100;
FOR j:= 1 TO NumberOfParties DO
   IF Distance [j].Magnitude < MinimalDistance THEN BEGIN
      MinimalDistance:= Distance [j].Magnitude;
      DistancePartyNumber:= j;

      DistanceConstituencyNumber:= 1;
      WHILE DistanceConstituencyNumber <> Distance [j].ConstituencyNumber DO
         DistanceConstituencyNumber:= DistanceConstituencyNumber + 1;
   END; {Then}

Adjustment [GlobalDistanceNumber].Distance := MinimalDistance;
Adjustment [GlobalDistanceNumber].PartyNumber := DistancePartyNumber;
Adjustment [GlobalDistanceNumber].ConstituencyNumber:=
   DistanceConstituencyNumber;
DistanceNumber [DistancePartyNumber] :=
   DistanceNumber [DistancePartyNumber] + 1;

END; {DeterminationOfMinimalDistances}

{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}
```

```
PROCEDURE DeterminationOfImprovement (VAR Improvement: ConstituencyVector;
                                      Representation:  ConstituencyVector);

BEGIN {DeterminationOfImprovement}

IF (RemainderMalRepresentation [DistanceConstituencyNumber] > 0) AND
   (GlobalDistanceNumber <= Representation [ActualConstituencyNumber])
   THEN BEGIN
   RemainderMalRepresentation [DistanceConstituencyNumber] :=
      RemainderMalRepresentation [DistanceConstituencyNumber] - 1;
   Improvement [ActualConstituencyNumber] :=
      Improvement [ActualConstituencyNumber] + 1;
   END; {Then}

END; {DeterminationOfImprovement}

{-------------------------------------------------------------------------------}
{-------------------------------------------------------------------------------}

PROCEDURE CalculationOfAdjustmentDistance
                               (VAR AdjustmentDistance: RealConstituencyVector;
                                Representation:         ConstituencyVector);

BEGIN {CalculationOfAdjustmentDistance}

AdjustmentDistance [ActualConstituencyNumber] :=
   AdjustmentWeight *
   Adjustment [Representation [ActualConstituencyNumber]].Distance  +
  (1 - AdjustmentWeight) *
   Adjustment [Representation [ActualConstituencyNumber] + 1].Distance;

END; {CalculationOfAdjustmentDistance}

{-------------------------------------------------------------------------------}
{-------------------------------------------------------------------------------}
```

```
PROCEDURE CalculationOfUpDistances;

VAR QuotientChallengingPartyRepr:  RealPartyVector;
    ChallengedPartyRepresentative: PartyAndBenchmarkRecordVector;

{Finds the (UnderRepresentation [i] + 1) best distances.}

BEGIN {CalculationOfUpDistances}

FOR j:= 1 TO NumberOfParties DO BEGIN

   DistanceNumber [j] := 1;
   DeterminationOfAssignedRepresen (ActualConstituencyNumber);

   QuotientChallengingPartyRepr [j] :=
      LN(ConstituencyMultiplier [ActualConstituencyNumber]) +
      DivisorQuotients [ActualConstituencyNumber, j,
                        (AssignedConstituencyRepr [j] + 1)];

   PositionWithinParty [j] := PartyRepresentatives [j];
   WHILE (SortedParties [PositionWithinParty [j], j].ConstituencyNumber =
      ActualConstituencyNumber) AND (PositionWithinParty [j] > 0) DO
      PositionWithinParty [j] := PositionWithinParty [j] - 1;

   ChallengedPartyRepresentative [j].Magnitude:=
      SortedParties [PositionWithinParty [j], j].Quotient;
   ChallengedPartyRepresentative [j].ConstituencyNumber:=
      SortedParties [PositionWithinParty [j], j].ConstituencyNumber;

   Distance [j].Magnitude :=
      ChallengedPartyRepresentative [j].Magnitude -
      QuotientChallengingPartyRepr [j];
   Distance [j].ConstituencyNumber :=
      ChallengedPartyRepresentative [j].ConstituencyNumber;

END; {For j}

GlobalDistanceNumber:= 1;
WHILE GlobalDistanceNumber <=
  (UnderRepresentation [ActualConstituencyNumber] + 1) DO BEGIN

   DeterminationOfMinimalDistances;
   IF s = 2 THEN
      DeterminationOfImprovement (UpImprovement, UnderRepresentation);

   QuotientChallengingPartyRepr [DistancePartyNumber] :=
      LN(ConstituencyMultiplier [ActualConstituencyNumber]) +
      DivisorQuotients [ActualConstituencyNumber, DistancePartyNumber,
     (AssignedConstituencyRepr [DistancePartyNumber] +
      DistanceNumber [DistancePartyNumber]) ];

   IF PositionWithinParty [DistancePartyNumber] > 0 THEN
      PositionWithinParty [DistancePartyNumber] :=
        (PositionWithinParty [DistancePartyNumber] - 1);
   WHILE (SortedParties [PositionWithinParty [DistancePartyNumber],
      DistancePartyNumber].ConstituencyNumber = ActualConstituencyNumber)
      AND (PositionWithinParty [DistancePartyNumber] > 0) DO
      PositionWithinParty [DistancePartyNumber] :=
        (PositionWithinParty [DistancePartyNumber] - 1);

   ChallengedPartyRepresentative [DistancePartyNumber].Magnitude:=
      SortedParties [PositionWithinParty [DistancePartyNumber],
                     DistancePartyNumber].Quotient;
   ChallengedPartyRepresentative [DistancePartyNumber].ConstituencyNumber:=
      SortedParties [PositionWithinParty [DistancePartyNumber],
                     DistancePartyNumber].ConstituencyNumber;

   Distance [DistancePartyNumber].Magnitude :=
      ChallengedPartyRepresentative [DistancePartyNumber].Magnitude -
      QuotientChallengingPartyRepr [DistancePartyNumber];
   Distance [DistancePartyNumber].ConstituencyNumber:=
      ChallengedPartyRepresentative [DistancePartyNumber].ConstituencyNumber;

   GlobalDistanceNumber:= GlobalDistanceNumber + 1;
END; {While}

CalculationOfAdjustmentDistance (Increase, UnderRepresentation);

END; {CalculationOfUpDistances}

{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}
```

```
PROCEDURE CalculationOfDownDistances;

VAR QuotientOutgoingPartyRepr:   RealPartyVector;
    IncomingPartyRepresentative: PartyAndBenchmarkRecordVector;

BEGIN {CalculationOfDownDistances}

FOR j:= 1 TO NumberOfParties DO BEGIN

   DistanceNumber [j] := 1;
   DeterminationOfAssignedRepresen (ActualConstituencyNumber);

   IF AssignedConstituencyRepr [j] > 0 THEN
      QuotientOutgoingPartyRepr [j] :=
         LN(ConstituencyMultiplier [ActualConstituencyNumber]) +
         DivisorQuotients [ActualConstituencyNumber, j,
                           AssignedConstituencyRepr [j]]
   ELSE QuotientOutgoingPartyRepr [j] := 50;

   PositionWithinParty [j] := PartyRepresentatives [j] + 1;
   WHILE (SortedParties [PositionWithinParty [j], j].ConstituencyNumber =
      ActualConstituencyNumber) AND
     (PositionWithinParty [j] < (NumberOfSeats + 1)) DO
      PositionWithinParty [j] := PositionWithinParty [j] + 1;

   IncomingPartyRepresentative [j].Magnitude:=
      SortedParties [PositionWithinParty [j], j].Quotient;
   IncomingPartyRepresentative [j].ConstituencyNumber :=
      SortedParties [PositionWithinParty [j], j].ConstituencyNumber;

   Distance [j].Magnitude:= QuotientOutgoingPartyRepr [j] -
      IncomingPartyRepresentative [j].Magnitude;
   Distance [j].ConstituencyNumber:=
      IncomingPartyRepresentative [j].ConstituencyNumber;

END; {For j}

GlobalDistanceNumber:= 1;
WHILE GlobalDistanceNumber <=
  (OverRepresentation [ActualConstituencyNumber] + 1) DO BEGIN

   DeterminationOfMinimalDistances;
   IF s = 2 THEN
      DeterminationOfImprovement (DownImprovement, OverRepresentation);

   IF (AssignedConstituencyRepr [DistancePartyNumber]
       + 1 - DistanceNumber [DistancePartyNumber]) > 0 THEN
      QuotientOutgoingPartyRepr [DistancePartyNumber] :=
         LN(ConstituencyMultiplier [ActualConstituencyNumber]) +
         DivisorQuotients [ActualConstituencyNumber, DistancePartyNumber,
        (AssignedConstituencyRepr [DistancePartyNumber]
         + 1 - DistanceNumber [DistancePartyNumber]) ]
   ELSE QuotientOutgoingPartyRepr [DistancePartyNumber] := 50;

   IF PositionWithinParty [DistancePartyNumber] < (NumberOfSeats + 1) THEN
      PositionWithinParty [DistancePartyNumber] :=
        (PositionWithinParty [DistancePartyNumber] + 1);
   WHILE (SortedParties [PositionWithinParty [DistancePartyNumber],
      DistancePartyNumber].ConstituencyNumber = ActualConstituencyNumber) AND
     (PositionWithinParty [DistancePartyNumber] < (NumberOfSeats + 1)) DO
      PositionWithinParty [DistancePartyNumber] :=
        (PositionWithinParty [DistancePartyNumber] + 1);

   IncomingPartyRepresentative [DistancePartyNumber].Magnitude:=
      SortedParties [PositionWithinParty [DistancePartyNumber],
                     DistancePartyNumber].Quotient;
   IncomingPartyRepresentative [DistancePartyNumber].ConstituencyNumber:=
      SortedParties [PositionWithinParty [DistancePartyNumber],
                     DistancePartyNumber].ConstituencyNumber;

   Distance [DistancePartyNumber].Magnitude:=
      QuotientOutgoingPartyRepr [DistancePartyNumber] -
      IncomingPartyRepresentative [DistancePartyNumber].Magnitude;
   Distance [DistancePartyNumber].ConstituencyNumber:=
      IncomingPartyRepresentative [DistancePartyNumber].ConstituencyNumber;

   GlobalDistanceNumber:= GlobalDistanceNumber + 1;
END; {While}

CalculationOfAdjustmentDistance (Decrease, OverRepresentation);

END; {CalculationOfDownDistances}

{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}
```

```
PROCEDURE AdjustmentOfMultiplier (Power: double);

BEGIN {AdjustmentOfMultiplier}

ConstituencyMultiplier [SelectedConstituencyNumber] :=
   ConstituencyMultiplier [SelectedConstituencyNumber] * EXP(Power);

END; {AdjustmentOfMultiplier}

{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}

PROCEDURE MaxSortingWithRestriction
                                    (VAR MaxItem:              integer;
                                     VAR ConstituencyNumber: integer;
                                     VAR SeveralWithMaxItem: boolean;
                                     Item:                   ConstituencyVector;
                                     MaxRestriction:         integer;
                                     RestrictionItem:        ConstituencyVector);

BEGIN {MaxSortingWithRestriction}

MaxItem:= 0;
SeveralWithMaxItem:= False;

FOR i:= 1 TO NumberOfConstituencies DO
   IF RestrictionItem [i] >= MaxRestriction THEN
      IF Item [i] > MaxItem THEN BEGIN
         MaxItem:= Item [i];
         ConstituencyNumber:= i;
         SeveralWithMaxItem:= False;
      END {Then}
      ELSE IF Item [i] = MaxItem THEN
              SeveralWithMaxItem:= True;

END; {MaxSortingWithRestriction}

{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}

PROCEDURE RotationOfConstituencies
                          (VAR RotationConstituencyNumber: integer;
                           MaxFirstCriterion:              integer;
                           MaxSecondCriterion:             integer;
                           FirstCriterion:                 ConstituencyVector;
                           SecondCriterion:                ConstituencyVector);

{Rotates the chosen malrepresented constituency, because it is important to
 vary. The corresponding procedure in ADJUSTMENT.PAS uses a WHILE loop.}

{In this program there are also some other differences regarding formulaes for
 Up and DownRotationConstituencyNumbers compared to ADJUSTMENT.PAS, including
 the initialization of the variable UpAdjustmentLastIteration.}

BEGIN {RotationOfConstituencies}

REPEAT
   IF RotationConstituencyNumber = NumberOfConstituencies THEN
      RotationConstituencyNumber:= 1
   ELSE RotationConstituencyNumber:= RotationConstituencyNumber + 1;
UNTIL (FirstCriterion [RotationConstituencyNumber] >= MaxFirstCriterion) AND
      (SecondCriterion [RotationConstituencyNumber] >= MaxSecondCriterion);

SelectedConstituencyNumber:= RotationConstituencyNumber;

END; {RotationOfConstituencies}

{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}
```

```
PROCEDURE RepresentationAdjustmentSelecti;

BEGIN {RepresentationAdjustmentSelection}

FOR i:= 1 TO NumberOfConstituencies DO BEGIN
   Increase [i] := 0;
   Decrease [i] := 0;
END; {For i}

MaxSortingWithRestriction
  (MaxUnderRepresentation, UnderReprConstituencyNumber,
   SeveralWithMaxUnderRepr, UnderRepresentation,
   1, UnderRepresentation);
MaxSortingWithRestriction
  (MaxOverRepresentation, OverReprConstituencyNumber,
   SeveralWithMaxOverRepr, OverRepresentation, 1, OverRepresentation);

{A non-binding restriction is included here, so we can utilize an already
 existing procedure, thereby reducing the number of procedures. The same
 comments apply to the utilization of the procedure RotationOfConstituencies
 below.}

IF MaxUnderRepresentation > MaxOverRepresentation THEN BEGIN
   IF NOT(SeveralWithMaxUnderRepr) THEN
      SelectedConstituencyNumber:= UnderReprConstituencyNumber
   ELSE RotationOfConstituencies (UpRotationConstituencyNumber,
                                  MaxUnderRepresentation, MaxUnderRepresentation,
                                  UnderRepresentation, UnderRepresentation);
   UpAdjustmentLastIteration:= True;
   ActualConstituencyNumber:= SelectedConstituencyNumber;
   CalculationOfUpDistances;
   AdjustmentOfMultiplier (Increase [SelectedConstituencyNumber]);
END; {Then}

IF MaxOverRepresentation > MaxUnderRepresentation THEN BEGIN
   IF NOT(SeveralWithMaxOverRepr) THEN
      SelectedConstituencyNumber:= OverReprConstituencyNumber
   ELSE RotationOfConstituencies (DownRotationConstituencyNumber,
                                  MaxOverRepresentation, MaxOverRepresentation,
                                  OverRepresentation, OverRepresentation);
   UpAdjustmentLastIteration:= False;
   ActualConstituencyNumber:= SelectedConstituencyNumber;
   CalculationOfDownDistances;
   AdjustmentOfMultiplier (-Decrease [SelectedConstituencyNumber]);
END; {Then}

IF MaxUnderRepresentation = MaxOverRepresentation THEN BEGIN
   IF UpAdjustmentLastIteration THEN BEGIN
      IF SeveralWithMaxOverRepr THEN
         RotationOfConstituencies (DownRotationConstituencyNumber,
                                  MaxOverRepresentation, MaxOverRepresentation,
                                  OverRepresentation, OverRepresentation)
      ELSE SelectedConstituencyNumber:= OverReprConstituencyNumber;
      UpAdjustmentLastIteration:= False;
      ActualConstituencyNumber:= SelectedConstituencyNumber;
      CalculationOfDownDistances;
      AdjustmentOfMultiplier (-Decrease [SelectedConstituencyNumber]);
   END {Then}
   ELSE BEGIN IF SeveralWithMaxUnderRepr THEN
                RotationOfConstituencies (UpRotationConstituencyNumber,
                                  MaxUnderRepresentation, MaxUnderRepresentation,
                                  UnderRepresentation, UnderRepresentation)
            ELSE SelectedConstituencyNumber:= UnderReprConstituencyNumber;
      UpAdjustmentLastIteration:= True;
      ActualConstituencyNumber:= SelectedConstituencyNumber;
      CalculationOfUpDistances;
      AdjustmentOfMultiplier (Increase [SelectedConstituencyNumber]);
   END; {Else}

END; {Then}

END; {RepresentationAdjustmentSelection}

{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}
```

```
PROCEDURE ImprovementAdjustmentSelection;

VAR MaxUpImprovement:                integer;
    MaxDownImprovement:              integer;
    UpImprConstituencyNumber:        integer;
    DownImprConstituencyNumber:      integer;
    SeveralWithMaxUpImprovement:     boolean;
    SeveralWithMaxDownImprovement: boolean;

BEGIN {ImprovementAdjustmentSelection}

FOR i:= 1 TO NumberOfConstituencies DO BEGIN
   UpImprovement [i] := 0;
   DownImprovement [i] := 0;
   Increase [i] := 0;
   Decrease [i] := 0;
END; {For i}

FOR ActualConstituencyNumber:= 1 TO NumberOfConstituencies DO
   IF UnderRepresentation [ActualConstituencyNumber] > 0 THEN BEGIN
      FOR i:= 1 TO NumberOfConstituencies DO
         RemainderMalRepresentation [i] := OverRepresentation [i];
      CalculationOfUpDistances;
   END {Then}
   ELSE IF OverRepresentation [ActualConstituencyNumber] > 0 THEN BEGIN
           FOR i:= 1 TO NumberOfConstituencies DO
              RemainderMalRepresentation [i] := UnderRepresentation [i];
           CalculationOfDownDistances;
        END; {Then}

MaxSortingWithRestriction (MaxUpImprovement, UpImprConstituencyNumber,
                           SeveralWithMaxUpImprovement, UpImprovement,
                           1, UnderRepresentation);

MaxSortingWithRestriction (MaxDownImprovement, DownImprConstituencyNumber,
                           SeveralWithMaxDownImprovement, DownImprovement,
                           1, OverRepresentation);

IF MaxUpImprovement > MaxDownImprovement THEN BEGIN
   IF SeveralWithMaxUpImprovement THEN BEGIN
      MaxSortingWithRestriction
                          (MaxUnderRepresentation, UnderReprConstituencyNumber,
                           SeveralWithMaxUnderRepr, UnderRepresentation,
                           MaxUpImprovement, UpImprovement);
      IF SeveralWithMaxUnderRepr THEN
         RotationOfConstituencies (UpRotationConstituencyNumber,
                                   MaxUpImprovement, MaxUnderRepresentation,
                                   UpImprovement, UnderRepresentation)
      ELSE SelectedConstituencyNumber:= UnderReprConstituencyNumber;
   END {Then}
   ELSE SelectedConstituencyNumber:= UpImprConstituencyNumber;
   UpAdjustmentLastIteration:= True;
   AdjustmentOfMultiplier (Increase [SelectedConstituencyNumber]);
END; {Then}

IF MaxDownImprovement > MaxUpImprovement THEN BEGIN
   IF SeveralWithMaxDownImprovement THEN BEGIN
      MaxSortingWithRestriction
                          (MaxOverRepresentation, OverReprConstituencyNumber,
                           SeveralWithMaxOverRepr, OverRepresentation,
                           MaxDownImprovement, DownImprovement);
      IF SeveralWithMaxOverRepr THEN
         RotationOfConstituencies (DownRotationConstituencyNumber,
                                   MaxDownImprovement, MaxOverRepresentation,
                                   DownImprovement, OverRepresentation)
      ELSE SelectedConstituencyNumber:= OverReprConstituencyNumber;
   END {Then}
   ELSE SelectedConstituencyNumber:= DownImprConstituencyNumber;
   UpAdjustmentLastIteration:= False;
   AdjustmentOfMultiplier (-Decrease [SelectedConstituencyNumber]);
END; {Then}
```

```
IF MaxUpImprovement = MaxDownImprovement THEN BEGIN

   MaxSortingWithRestriction
                          (MaxUnderRepresentation, UnderReprConstituencyNumber,
                           SeveralWithMaxUnderRepr, UnderRepresentation,
                           MaxUpImprovement, UpImprovement);

   MaxSortingWithRestriction
                           (MaxOverRepresentation, OverReprConstituencyNumber,
                            SeveralWithMaxOverRepr, OverRepresentation,
                            MaxDownImprovement, DownImprovement);

   IF MaxUnderRepresentation > MaxOverRepresentation THEN BEGIN
      IF SeveralWithMaxUnderRepr THEN
         RotationOfConstituencies (UpRotationConstituencyNumber,
                                   MaxUpImprovement, MaxUnderRepresentation,
                                   UpImprovement, UnderRepresentation)
      ELSE SelectedConstituencyNumber:= UnderReprConstituencyNumber;
      UpAdjustmentLastIteration:= True;
      AdjustmentOfMultiplier (Increase [SelectedConstituencyNumber]);
   END; {Then}

   IF MaxOverRepresentation > MaxUnderRepresentation THEN BEGIN
      IF SeveralWithMaxOverRepr THEN
         RotationOfConstituencies (DownRotationConstituencyNumber,
                                   MaxDownImprovement, MaxOverRepresentation,
                                   DownImprovement, OverRepresentation)
      ELSE SelectedConstituencyNumber:= OverReprConstituencyNumber;
      UpAdjustmentLastIteration:= False;
      AdjustmentOfMultiplier (-Decrease [SelectedConstituencyNumber]);
   END; {Then}

   IF MaxUnderRepresentation = MaxOverRepresentation THEN
      IF UpAdjustmentLastIteration THEN BEGIN
         IF SeveralWithMaxDownImprovement THEN
            RotationOfConstituencies (DownRotationConstituencyNumber,
                                      MaxDownImprovement, MaxOverRepresentation,
                                      DownImprovement, OverRepresentation)
         ELSE SelectedConstituencyNumber:= DownImprConstituencyNumber;
         UpAdjustmentLastIteration:= False;
         AdjustmentOfMultiplier (-Decrease [SelectedConstituencyNumber]);
      END {Then}
      ELSE BEGIN
         IF SeveralWithMaxUpImprovement THEN
            RotationOfConstituencies (UpRotationConstituencyNumber,
                                      MaxUpImprovement, MaxUnderRepresentation,
                                      UpImprovement, UnderRepresentation)
         ELSE SelectedConstituencyNumber:= UpImprConstituencyNumber;
         UpAdjustmentLastIteration:= True;
         AdjustmentOfMultiplier (Increase [SelectedConstituencyNumber]);
      END; {Else}

END; {Then}

END; {ImprovementAdjustmentSelection}

{-------------------------------------------------------------------------------}
{-------------------------------------------------------------------------------}

PROCEDURE UpdatingOfUnsortedParties;

{Updates quotients for the constituency whose multiplier has been adjusted.}

BEGIN {UpdatingOfUnsortedParties}

FOR j:= 1 TO NumberOfParties DO
   FOR k:= 1 TO NumberOfDivisors DO
      UnsortedParties [SelectedConstituencyNumber, j, k].Quotient:=
         LN(ConstituencyMultiplier [SelectedConstituencyNumber]) +
         DivisorQuotients [SelectedConstituencyNumber, j, k];

END; {UpdatingOfUnsortedParties}

{-------------------------------------------------------------------------------}
{-------------------------------------------------------------------------------}
```

```
PROCEDURE Iteration;

{This procedure puts together the procedures in the solution section}

BEGIN {Iteration}

SortingWithinTheParties;
MeasureOfGoodnessCalculation;

IF NumberOfIterations = 0 THEN
   InitialMeasureOfGoodness:= MeasureOfGoodness;

IF (MeasureOfGoodness > 0) AND NotAbandoned THEN BEGIN
   CASE s OF
      1: RepresentationAdjustmentSelecti;
      2: ImprovementAdjustmentSelection;
   END; {Case s}

   UpdatingOfUnsortedParties;
   NumberOfIterations:= NumberOfIterations + 1;

   IF NumberOfIterations = MaxIterations THEN BEGIN
      writeln (Result, 'Maximum iterations, abandonment');
      writeln (Result);
      NotAbandoned:= False;
      Failures [s,n,c,p] := 1;
   END; {Then}

END {Then}
ELSE IF MeasureOfGoodness = 0 THEN
        NotFinished:= False;

END; {Iteration}

{--------------------------------------------------------------------------}
{--------------------------------------------------------------------------}

PROCEDURE CountryAssignmentAndSorting;

VAR PlacedSeats:         integer;
    PositionWithinParty: integer;
    ActualRecordNumber:  integer;

BEGIN {CountryAssignmentAndSorting}

FOR l:= 1 TO NumberOfSeats DO
   SortedCountry [l].Quotient:= -50;

FOR i:= 1 TO NumberOfConstituencies DO
   FOR j:= 1 TO NumberOfParties DO
      Representatives [i,j] := 0;

PlacedSeats:= 0;
PositionWithinParty:= 1;
FOR j:= 1 TO NumberOfParties DO BEGIN
   ActualRecordNumber:= 1;
   WHILE PositionWithinParty <= PartyRepresentatives [j] DO BEGIN

      WHILE SortedParties [PositionWithinParty, j].Quotient <
         SortedCountry [ActualRecordNumber].Quotient DO
         ActualRecordNumber:= ActualRecordNumber + 1;

      FOR l:= PlacedSeats DOWNTO ActualRecordNumber DO
         SortedCountry [l + 1] := SortedCountry [l];

      PlacedSeats:= PlacedSeats + 1;
      SortedCountry [ActualRecordNumber].ConstituencyNumber:=
         SortedParties [PositionWithinParty, j].ConstituencyNumber;
      SortedCountry [ActualRecordNumber].PartyNumber:= j;
      SortedCountry [ActualRecordNumber].Representative:=
         SortedParties [PositionWithinParty, j].Representative;
      SortedCountry [ActualRecordNumber].Quotient :=
         SortedParties [PositionWithinParty, j].Quotient;

      i:= 1;
      WHILE i <> SortedCountry [ActualRecordNumber].ConstituencyNumber DO
         i:= i + 1;
      Representatives [i,j] :=
         SortedCountry [ActualRecordNumber].Representative;

      PositionWithinParty:= PositionWithinParty + 1;
      ActualRecordNumber:= ActualRecordNumber + 1;
   END; {While}

   PositionWithinParty:= 1;
END; {For j}

END; {CountryAssignmentAndSorting}

{--------------------------------------------------------------------------}
{--------------------------------------------------------------------------}
```

```
PROCEDURE OutputOfMultipliers (ActualMultiplier: MultiplierMatrix);

BEGIN {OutputOfMultipliers}

writeln (Result);
FOR i:= 1 TO NumberOfConstituencies DO BEGIN
   write (Result, ConstituencyName [i], '      ');
   FOR f:= 1 TO 3 DO
      write (Result, ActualMultiplier [i,f,c,p] :8:5, '        ');
   writeln (Result);
END; {For i}
writeln (Result);

END; {OutputOfMultipliers}

{-------------------------------------------------------------------------------}
{-------------------------------------------------------------------------------}

PROCEDURE AlgorithmStatistics;

{Calculates and outputs algorithm statistics about iterations and measure of
 goodness.}

TYPE IterationVector =     ARRAY [1..NumberOfMatrixMethods] OF integer;
     RealIterationVector = ARRAY [1..NumberOfMatrixMethods] OF double;

VAR NumberOfInitialSolutions:        integer;
    SumIterations:                   IterationVector;
    SumFailures:                     IterationVector;
    SumGoodness:                     IterationVector;
    InitialSolutions:                IterationVector;
    AverageIterations:               RealIterationVector;
    AverageFailures:                 RealIterationVector;
    AverageGoodness:                 RealIterationVector;
    SumGoodnessToIterationRatio:     RealIterationVector;
    AverageGoodnessToIterationRatio: RealIterationVector;

BEGIN {AlgorithmStatistics}

writeln (Result, 'Statistics about initial measures of goodness ');
writeln (Result);

write (Result, 'Whole data set');
SumGoodness [1] := 0;
FOR n:= 1 TO 3 DO
   FOR c:= 1 TO NumberOfConstraintSets DO
      FOR p:= 1 TO NumberOfMatrixMethods DO
         SumGoodness [1] := SumGoodness [1] + Goodness [n,c,p];

   AverageGoodness [1] := SumGoodness [1] /
      (3 * NumberOfConstraintSets * NumberOfMatrixMethods);
   writeln (Result, '   Sum', SumGoodness [1] :5, ' Average',
      AverageGoodness [1] :6:2);
writeln (Result);

writeln (Result, 'Initialization ');
FOR n:= 1 TO 3 DO BEGIN
   SumGoodness [n] := 0;
   FOR c:= 1 TO NumberOfConstraintSets DO
      FOR p:= 1 TO NumberOfMatrixMethods DO
         SumGoodness [n] := SumGoodness [n] + Goodness [n,c,p];

   AverageGoodness [n] := SumGoodness [n] /
      (NumberOfConstraintSets * NumberOfMatrixMethods);
   CASE n OF
      1: write (Result, 'No           ');
      2: write (Result, 'Quota ratio  ');
      3: write (Result, 'Apportionment ');
   END; {Case n}
   writeln (Result, '   Sum', SumGoodness [n] :5, ' Average',
      AverageGoodness [n] :6:2);
END; {For n}
writeln (Result);
```

```
writeln (Result, 'Constraint set ');
FOR c:= 1 TO NumberOfConstraintSets DO BEGIN
   SumGoodness [c] := 0;
   FOR n:= 1 TO 3 DO
      FOR p:= 1 TO NumberOfMatrixMethods DO
         SumGoodness [c] := SumGoodness [c] + Goodness [n,c,p];

   AverageGoodness [c] := SumGoodness [c] / (3 * NumberOfMatrixMethods);
   writeln (Result, c:1, '                Sum', SumGoodness [c] :5,
      ' Average', AverageGoodness [c] :6:2);
END; {For c}
writeln (Result);

writeln (Result, 'Matrix method ');
FOR p:= 1 TO NumberOfMatrixMethods DO BEGIN
   SumGoodness [p] := 0;
   FOR n:= 1 TO 3 DO
      FOR c:= 1 TO NumberOfConstraintSets DO
         SumGoodness [p] := SumGoodness [p] + Goodness [n,c,p];

   AverageGoodness [p] := SumGoodness [p] / (3 * NumberOfConstraintSets);
   writeln (Result, MethodName [(p + 2)], '              Sum',
      SumGoodness [p] :5, ' Average', AverageGoodness [p] :6:2);
END; {For p}
writeln (Result);
WritingOfLine (Result);

writeln (Result);
writeln (Result, 'Statistics about number of iterations ');
writeln (Result);
writeln (Result, 'Selection ');
FOR s:= 1 TO 2 DO BEGIN
   SumIterations [s] := 0;
   SumFailures [s] := 0;
   FOR n:= 1 TO 3 DO
      FOR c:= 1 TO NumberOfConstraintSets DO
         FOR p:= 1 TO NumberOfMatrixMethods DO BEGIN
            SumIterations [s]:= SumIterations [s] + Iterations [s,n,c,p];
            SumFailures [s] := SumFailures [s] + Failures [s,n,c,p];
         END; {For p}

   AverageIterations [s] := SumIterations [s] /
      (3 * NumberOfConstraintSets * NumberOfMatrixMethods);
   AverageFailures [s] := SumFailures [s] /
      (3 * NumberOfConstraintSets * NumberOfMatrixMethods);
   IF s = 1 THEN
      write (Result, 'Representation ')
   ELSE write (Result, 'Improvement    ');
   writeln (Result, '  Sum', SumIterations [s] :5, ' Average',
      AverageIterations [s] :6:2, '   Sum abandonments ',
      SumFailures [s] :2, ' Average ',
      AverageFailures [s] :4:2);
END; {For s}
writeln (Result);

writeln (Result, 'Initialization ');
FOR n:= 1 TO 3 DO BEGIN
   SumIterations [n] := 0;
   SumFailures [n] := 0;
   FOR s:= 1 TO 2 DO
      FOR c:= 1 TO NumberOfConstraintSets DO
         FOR p:= 1 TO NumberOfMatrixMethods DO BEGIN
            SumIterations [n] := SumIterations [n] + Iterations [s,n,c,p];
            SumFailures [n] := SumFailures [n] + Failures [s,n,c,p];
         END; {For p}

   AverageIterations [n] := SumIterations [n] /
      (2 * NumberOfConstraintSets * NumberOfMatrixMethods);
   AverageFailures [n] := SumFailures [n] /
      (2 * NumberOfConstraintSets * NumberOfMatrixMethods);
   CASE n OF
      1: write (Result, 'No            ');
      2: write (Result, 'Quota ratio   ');
      3: write (Result, 'Apportionment ');
   END; {Case n}
   writeln (Result, '   Sum', SumIterations [n] :5, ' Average',
      AverageIterations [n] :6:2, '   Sum abandonments ',
      SumFailures [n] :2, ' Average ', AverageFailures [n] :4:2);
END; {For n}
writeln (Result);
```

```
writeln (Result, 'Constraint set ');
FOR c:= 1 TO NumberOfConstraintSets DO BEGIN
   SumIterations [c] := 0;
   SumFailures [c] := 0;
   FOR s:= 1 TO 2 DO
      FOR n:= 1 TO 3 DO
         FOR p:= 1 TO NumberOfMatrixMethods DO BEGIN
            SumIterations [c] := SumIterations [c] + Iterations [s,n,c,p];
            SumFailures [c] := SumFailures [c] + Failures [s,n,c,p];
         END; {For p}

   AverageIterations [c] := SumIterations [c] /
     (2 * 3 * NumberOfMatrixMethods);
   AverageFailures [c] := SumFailures [c] /
     (2 * 3 * NumberOfMatrixMethods);
   writeln (Result, c:1, '                    Sum', SumIterations [c] :5,
      ' Average', AverageIterations [c] :6:2, '   Sum abandonments ',
      SumFailures [c] :2, ' Average ', AverageFailures [c] :4:2);
END; {For c}
writeln (Result);

writeln (Result, 'Matrix method ');
FOR p:= 1 TO NumberOfMatrixMethods DO BEGIN
   SumIterations [p] := 0;
   SumFailures [p] := 0;
   FOR s:= 1 TO 2 DO
      FOR n:= 1 TO 3 DO
         FOR c:= 1 TO NumberOfConstraintSets DO BEGIN
            SumIterations [p] := SumIterations [p] + Iterations [s,n,c,p];
            SumFailures [p] := SumFailures [p] + Failures [s,n,c,p];
         END; {For p}

   AverageIterations [p] := SumIterations [p] /
     (2 * 3 * NumberOfConstraintSets);
   AverageFailures [p] := SumFailures [p] /
     (2 * 3 * NumberOfConstraintSets);
   writeln (Result, MethodName [(p + 2)], '              Sum',
      SumIterations [p] :5, ' Average', AverageIterations [p] :6:2,
      '   Sum abandonments ', SumFailures [p] :2, ' Average ',
      AverageFailures [p] :4:2);
END; {For p}
writeln (Result);
WritingOfLine (Result);

writeln (Result);
writeln (Result, 'Initial optimal solutions ');
writeln (Result);

NumberOfInitialSolutions:= 0;
FOR n:= 1 TO 3 DO BEGIN
   InitialSolutions [n] := 0;
   FOR c:= 1 TO NumberOfConstraintSets DO
      FOR p:= 1 TO NumberOfMatrixMethods DO
         IF Goodness [n,c,p] = 0 THEN
            InitialSolutions [n] := InitialSolutions [n] + 1;

   CASE n OF
      1: write (Result, 'No initialization            ');
      2: write (Result, 'Quota ratio initialization   ');
      3: write (Result, 'Apportionment initialization ');
   END; {Case n}
   writeln (Result, InitialSolutions [n] :3);
END; {For n}

writeln (Result);
writeln (Result, 'Average "goodness to iteration" ratio');
writeln (Result);
FOR s:= 1 TO 2 DO BEGIN
   SumGoodnessToIterationRatio [s] := 0;
   FOR n:= 1 TO 3 DO
      FOR c:= 1 TO NumberOfConstraintSets DO
         FOR p:= 1 TO NumberOfMatrixMethods DO
            IF Goodness [n,c,p] > 0 THEN
               SumGoodnessToIterationRatio [s]:=
                  SumGoodnessToIterationRatio [s] +
                 (Goodness [n,c,p] / Iterations [s,n,c,p]);

   AverageGoodnessToIterationRatio [s] := SumGoodnessToIterationRatio [s] /
      ((3 * NumberOfConstraintSets * NumberOfMatrixMethods)
      - NumberOfInitialSolutions);
   CASE s OF
      1: write (Result, 'Representation selection     ');
      2: write (Result, 'Improvement selection        ');
   END; {Case s}
   writeln (Result, AverageGoodnessToIterationRatio [s] :6:2);
END; {For s}
writeln (Result);
WritingOfLine (Result);
```

```
writeln (Result);
writeln (Result, 'Measures of goodness ');
writeln (Result);
FOR n:= 1 TO 3 DO BEGIN
   CASE n OF
      1: writeln (Result, 'No initialization ');
      2: writeln (Result, 'Quota ratio initialization ');
      3: writeln (Result, 'Apportionment initialization ');
   END; {Case n}
   writeln (Result);
   write (Result, '  ');
   FOR p:= 1 TO NumberOfMatrixMethods DO
      write (Result, MethodName [(p + 2)] :4);
   writeln (Result);
   FOR c:= 1 TO NumberOfConstraintSets DO BEGIN
      write (Result, c:1);
      FOR p:= 1 TO NumberOfMatrixMethods DO
         write (Result, Goodness [n,c,p] :4);
      writeln (Result);
   END; {For c}
   writeln (Result);
END; {For n}

FOR s:= 1 TO 2 DO BEGIN
   WritingOfLine (Result);
   writeln (Result);
   IF s = 1 THEN
      write (Result, 'Representation ')
   ELSE write (Result, 'Improvement ');
   writeln (Result, 'selection');
   writeln (Result);
   FOR n:= 1 TO 3 DO BEGIN
      CASE n OF
         1: writeln (Result, 'No initialization ');
         2: writeln (Result, 'Quota ratio initialization ');
         3: writeln (Result, 'Apportionment initialization ');
      END; {Case n}
      writeln (Result);
      write (Result, '    ');
      FOR p:= 1 TO NumberOfMatrixMethods DO
         write (Result, MethodName [(p + 2)] :4);
      writeln (Result);
      FOR c:= 1 TO NumberOfConstraintSets DO BEGIN
         write (Result, c:1, '  ');
         FOR p:= 1 TO NumberOfMatrixMethods DO
            write (Result, Iterations [s,n,c,p] :4);
         writeln (Result);
      END; {For c}
      writeln (Result);
   END; {For n}
END; {For s}

WritingOfLine (Result);
WritingOfLine (Result);

END; {AlgorithmStatistics}

{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}
```

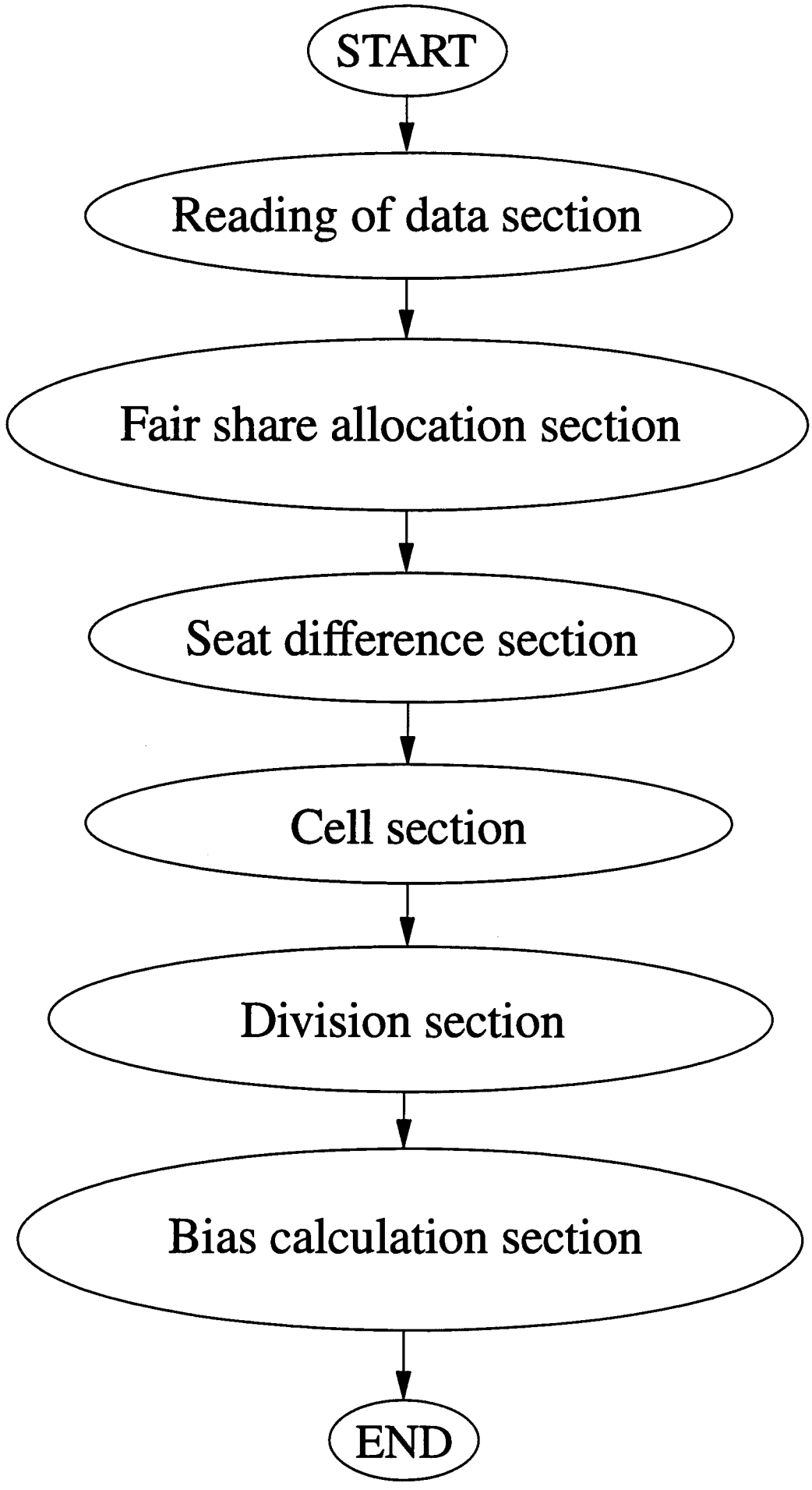
```
BEGIN {ElectionAlgorithm}

{Section for vector apportionment of constituencies and parties.}

ReadingOfElectionData (Ice63);

rewrite (Result);

CheckOfHouseSize (Result);

writeln (Result, 'Inputfile Ice63');
writeln (Result);
writeln (Result, 'Votes for the parties in the constituencies ');
OutputOfIntegerMatrices (Result, 8, 10, Votes);
WritingOfLine (Result);
WritingOfLine (Result);
writeln (Result);

MethodName [1] := 'EL ';
MethodName [2] := 'LF ';
MethodName [3] := 'SD ';
MethodName [4] := 'DM ';
MethodName [5] := 'MF ';
MethodName [6] := 'HA ';

CalculationOfSums (Votes, ConstituencyVotes, PartyVotes, SumVotes);

o:= 1;
FOR i:= 1 TO NumberOfConstituencies DO BEGIN
   ObjectVotes [o] := ConstituencyVotes [i];
   ObjectName [o] := ConstituencyName [i];
   o:= o + 1;
END; {For i}

FOR j:= 1 TO NumberOfParties DO BEGIN
   ObjectName [o]:= PartyName [j];
   ObjectVotes [o]:= PartyVotes [j];
   o:= o + 1;
END; {For j}

m:= 2;
WHILE m <= NumberOfVectorMethods DO BEGIN

   CASE m OF
      2: BEGIN
            Quota:= SumVotes / NumberOfSeats;
            QuotaMethod:= True;
            DivisorMethod:= False;
         END; {m = 2}
      3: Parameter:= 0;
      4: Parameter:= 1/3;
      5: Parameter:= 0.5;
      6: Parameter:= 1;
   END; {Case}

   IF m > 2 THEN BEGIN
      ParametricDivisorMethod;
      DivisorMethod:= True;
      QuotaMethod:= False;
   END; {Then}

   VectorApportionment (1, NumberOfConstituencies);
   VectorApportionment ((NumberOfConstituencies + 1), NumberOfObjects);

   m:= m + 1;
END; {While}

writeln (Result, 'The vector apportionment for the constituencies');
OutputOfVectorApportionment (Result, 1, NumberOfConstituencies);
writeln (Result, 'The vector apportionment for the parties');
OutputOfVectorApportionment (Result, (NumberOfConstituencies + 1),
                                     NumberOfObjects);

WritingOfLine (Result);
WritingOfLine (Result);

{------------------------------------------------------------------------------}
```

```
{Section for output to gamsfile for two dimensional LF apportionment}

rewrite (TwoDimLFGams);
QuotaViolation:= False;

FOR i:= 1 TO NumberOfConstituencies DO
   FOR j:= 1 TO NumberOfParties DO BEGIN
      IntegerParts [i,j] := (Votes [i,j] * NumberOfSeats) DIV SumVotes;
      Remainders [i,j] := (Votes [i,j] * NumberOfSeats) MOD SumVotes;
      RelevantCosts [i,j] := SumVotes - (2 * Remainders [i,j]);
   END; {For j}

FOR i:= 1 TO NumberOfConstituencies DO
   FOR j:= 1 TO NumberOfParties DO
      WholeSeats [i,j] := IntegerParts [i,j];

CalculationOfSums (WholeSeats, WholeConstituencySeats,
                   WholePartySeats, SumWholeSeats);

FOR i:= 1 TO NumberOfConstituencies DO BEGIN
   FractionalConstituencySeats [i] :=
      VectorRepresentatives [i,5] - WholeConstituencySeats [i];
   IF FractionalConstituencySeats [i] < 0 THEN
      QuotaViolation:= True;
END; {For i}

FOR j:= 1 TO NumberOfParties DO BEGIN
   FractionalPartySeats [j] :=
      VectorRepresentatives [(j + NumberOfConstituencies), 5]
         - WholePartySeats [j];
   IF FractionalPartySeats [j] < 0 THEN
      QuotaViolation:= True;
END; {For j}

{It is necessary to check for quota violation because divisor methods do not
 "stay within the quota".}

IF QuotaViolation THEN BEGIN
   writeln (Result, 'Quota violation ');
   WritingOfLine (Result);
   writeln (Result);
END {Then}
ELSE OutputToTwoDimLFGamsFile (TwoDimLFGams);

{-------------------------------------------------------------------------------}

{Section for multiplier initializations}

rewrite (BiasApport);

writeln (Result);
writeln (Result, 'The constraint sets are  1: EL-EL,  2: MF-MF,  3: SD-HA ');
writeln (Result);
WritingOfLine (Result);
WritingOfLine (Result);
writeln (Result);

FOR c:= 1 TO NumberOfConstraintSets DO BEGIN

   FOR i:= 1 TO NumberOfConstituencies DO
      CASE c OF
         1: ConstituencyRepresentatives [i] := VectorRepresentatives [i,1];
         2: ConstituencyRepresentatives [i] := VectorRepresentatives [i,5];
         3: ConstituencyRepresentatives [i] := VectorRepresentatives [i,3];
      END; {Case c}

   FOR j:= 1 TO NumberOfParties DO
      CASE c OF
         1: PartyRepresentatives [j] :=
               VectorRepresentatives [(j + NumberOfConstituencies), 1];
         2: PartyRepresentatives [j] :=
               VectorRepresentatives [(j + NumberOfConstituencies), 5];
         3: PartyRepresentatives [j] :=
               VectorRepresentatives [(j + NumberOfConstituencies), 6];
      END; {Case c}
```

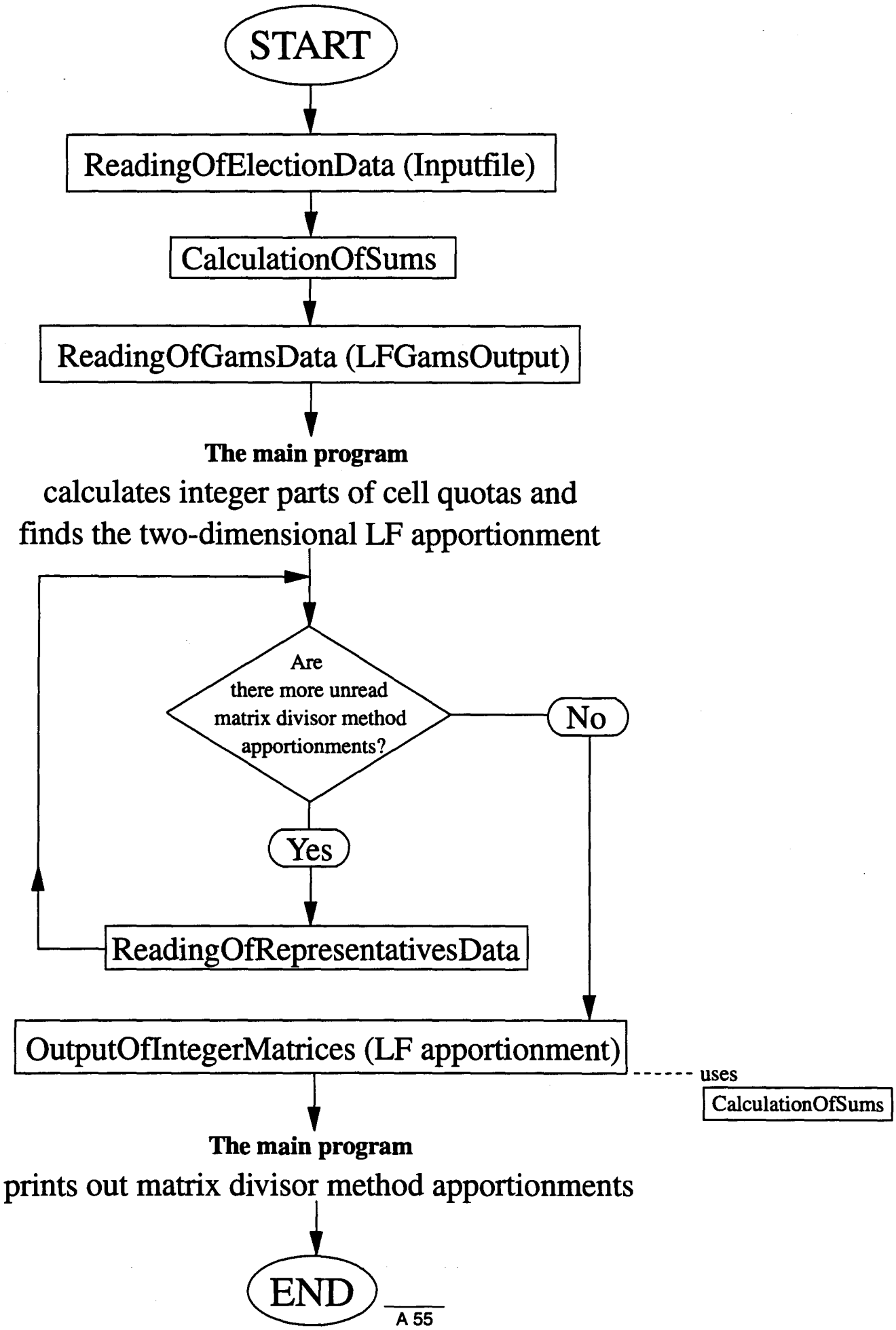
```
   CalculationOfQuotas;

   FOR p:= 1 TO NumberOfMatrixMethods DO BEGIN

      CASE p OF
         1: Parameter:= 0.01;
         2: Parameter:= 1/3;
         3: Parameter:= 0.5;
         4: Parameter:= 1;
      END; {Case}

      ParametricDivisorMethod;
      InitializeDivisorQuotients;

      n:= 1;
      WHILE n <= 3 DO BEGIN

         CASE n OF
            1: NoInitialization;
            2: QuotaRatioInitialization;
            3: ApportionmentInitialization;
         END; {Case}

         FOR i:= 1 TO NumberOfConstituencies DO
            InitialMultiplier [i] := ConstituencyMultiplier [i];

{------------------------------------------------------------------------------}

{Section for solving the matrix problem}

         UpRotationConstituencyNumber:= NumberOfConstituencies;
         DownRotationConstituencyNumber:= NumberOfConstituencies;

         FOR j:= 1 TO NumberOfParties DO BEGIN
            SortedParties [0,j].Quotient := 50;
            SortedParties [0,j].ConstituencyNumber:= NumberOfConstituencies;
            SortedParties [(NumberOfSeats + 1) ,j].Quotient := -50;
         END; {For j}

         s:= 1;
         WHILE s <= 2 DO BEGIN
            NotFinished:= True;
            NotAbandoned:= True;
            UpAdjustmentLastIteration:= True;
            NumberOfIterations:= 0;

            InitializationOfUnsortedParties;

            WHILE NotFinished AND NotAbandoned DO
               Iteration;

            Iterations [s,n,c,p] := NumberOfIterations;

            IF s = 1 THEN
               FOR i:= 1 TO NumberOfConstituencies DO
                  ConstituencyMultiplier [i] := InitialMultiplier [i];

            s:= s + 1;
         END; {While s}

         Goodness [n,c,p] := InitialMeasureOfGoodness;

         n:= n + 1;
      END; {While n}

      MultiplicativeNormalization (NormalizationFactor [3,c,p]);

      FOR i:= 1 TO NumberOfConstituencies DO BEGIN
         ConstituencyMultiplier [i] :=
            ConstituencyMultiplier [i] * NormalizationFactor [3,c,p];
         Multiplier [i,3,c,p]:= ConstituencyMultiplier [i];
      END; {For i}

      FOR i:= 1 TO NumberOfConstituencies DO
         FOR j:= 1 TO NumberOfParties DO
            Apportionment [i,j,c,p] := Representatives [i,j];

      FOR i:= 1 TO NumberOfConstituencies DO
         Components [i,1,c,p] := Multiplier [i,1,c,p];

      FOR MultiplierNumber:= 2 TO 3 DO BEGIN
         FOR i:= 1 TO NumberOfConstituencies DO
            Components [i, MultiplierNumber, c, p] :=
               Multiplier [i, MultiplierNumber, c, p] /
               Multiplier [i, (MultiplierNumber - 1), c, p];
      END; {For MultiplierNumber}
```

```
      CountryAssignmentAndSorting;

      IF NotAbandoned THEN BEGIN
         writeln (Result, 'The matrix apportionment with constraint set ', c:1,
            ' and ', MethodName [(p + 2)]);
         OutputOfIntegerMatrices (Result, 5, 6, Representatives);

         WritingOfLine (Result);
         writeln (Result);
         writeln (Result, 'Normalization factors and normalized constituency ',
            'multipliers with constraint');
         writeln (Result, 'set ', c:1, ' and ', MethodName [(p + 2)]);
         writeln (Result);
         write (Result, 'Factors  ');
         FOR f:= 1 TO 3 DO
            write (Result, NormalizationFactor [f,c,p] :8:5, '        ');
         writeln (Result);
         writeln (Result);
         writeln (Result, '           Ratio       Apportionment     Suitable');
         OutputOfMultipliers (Multiplier);

         write (Result, 'The (logarithmic) marginal value of representation ');
         writeln (Result, 'is ', MarginalValueOfRepresentation [c,p] :6:3,
            ' while');
         write (Result, 'the (logarithmic) initial marginal value of ');
         write (Result, 'representation is ');
         writeln (Result, LN(InitialValueOfRepresentation):6:3, '.');
         writeln (Result);
         write (Result, 'The normalization factor for apportionment ');
         writeln (Result, 'initialization with ', MethodName [p + 2],
            'represents');
         write (Result, 'a ', ((NormalizationFactor [2,c,p] - 1) * 100):6:2,
            ' % change compared to the initial marginal value of ');
         writeln (Result, 'representation');
         writeln (Result, 'with a target weight of ', TargetWeight:3:1, '.');
         writeln (Result);

         WritingOfLine (Result);
         writeln (Result);
         writeln (Result, 'Normalization factors and components with ',
            'constraint set ', c:1, ' and ', MethodName [(p + 2)]);
         writeln (Result);
         write (Result, 'Factors  ');
         FOR f:= 1 TO 3 DO
            write (Result, NormalizationFactor [f,c,p] :8:5, '        ');
         writeln (Result);
         writeln (Result);
         writeln (Result, '           Bound       Constituency       Matrix');
         OutputOfMultipliers (Components);
      END; {Then}
      WritingOfLine (Result);
      WritingOfLine (Result);
      writeln (Result);

      IF c = 2 THEN BEGIN
         writeln (BiasApport);
         writeln (BiasApport, 'Matrix apportionment with ',
            MethodName [(p + 2)]);
         IF NotAbandoned THEN
            OutputOfIntegerMatrices (BiasApport, 5, 6, Representatives)
         ELSE writeln (BiasApport, 'Abandonement ');
      END; {Then}

   END; {For p}
END; {For c}

AlgorithmStatistics;

END. {ElectionAlgorithm}
```

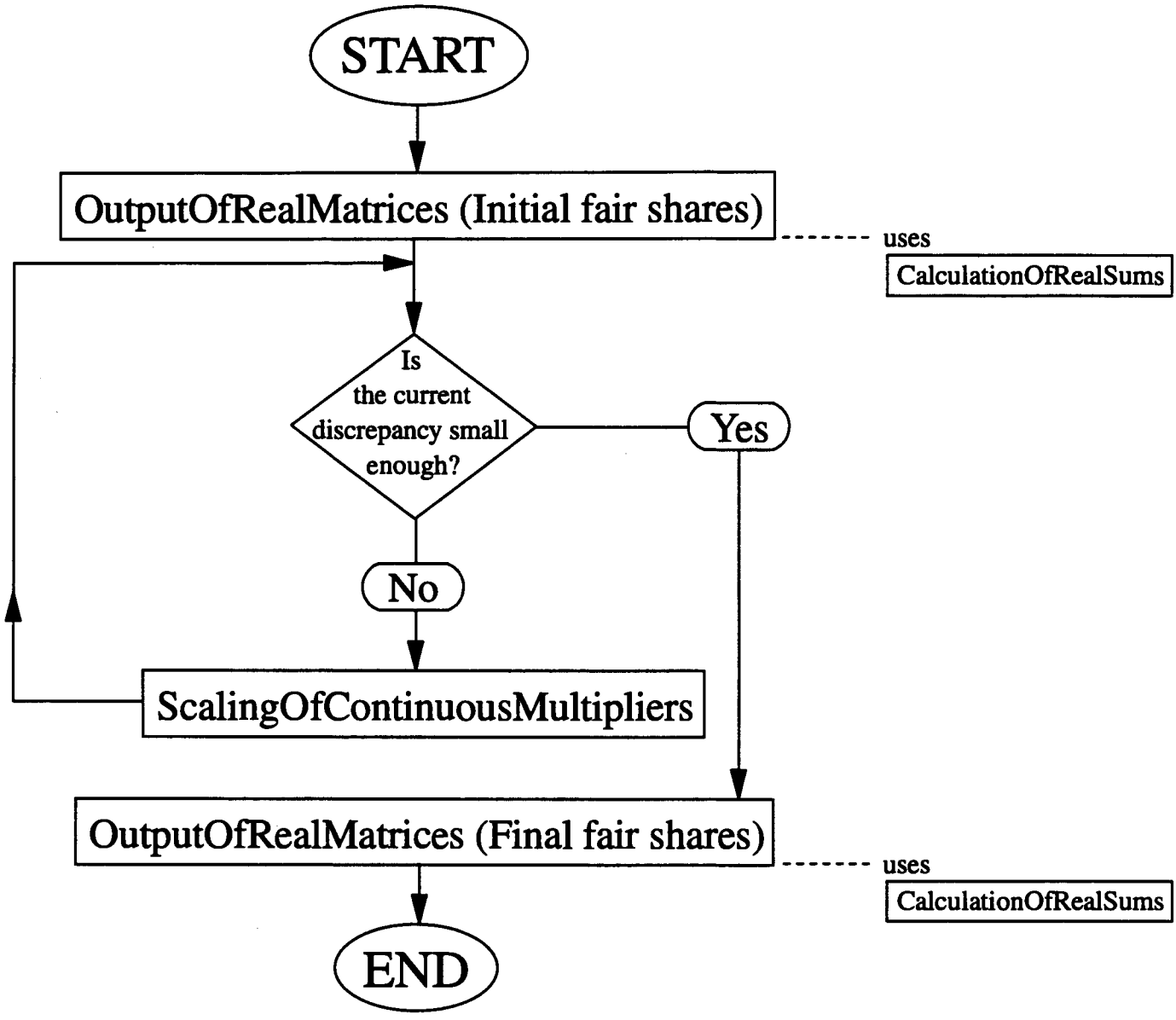
The MATRIX BIAS program



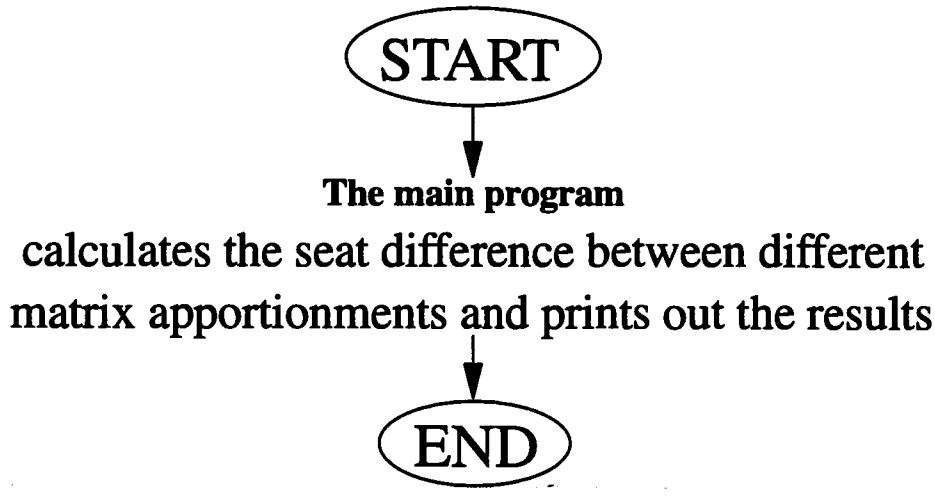
Reading of data section



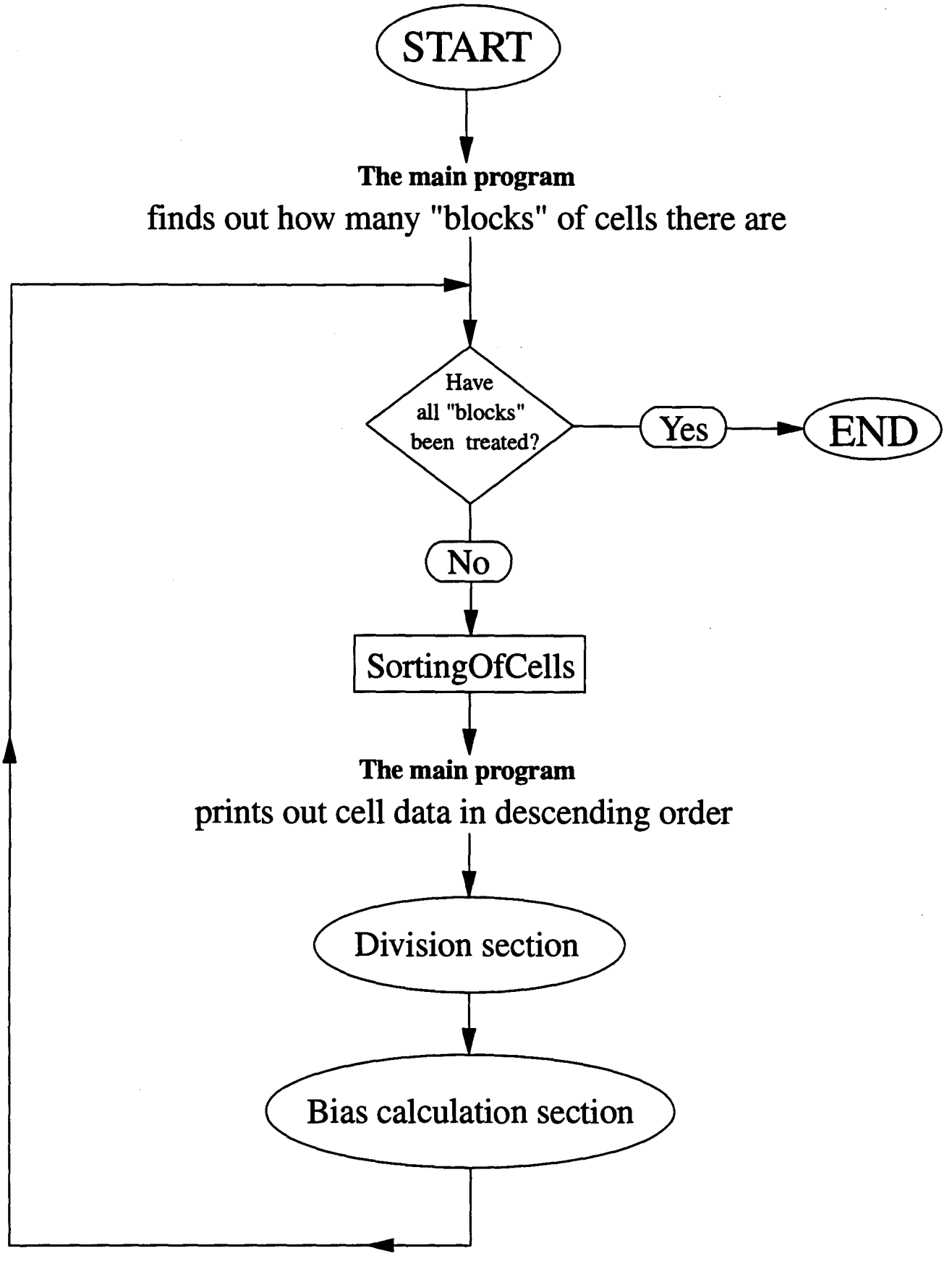
Fair share allocation section



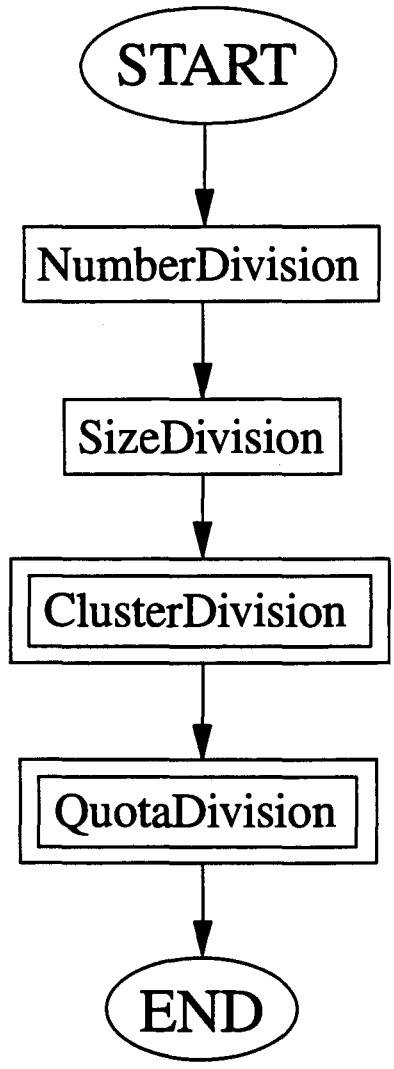
Seat difference section



Cell section

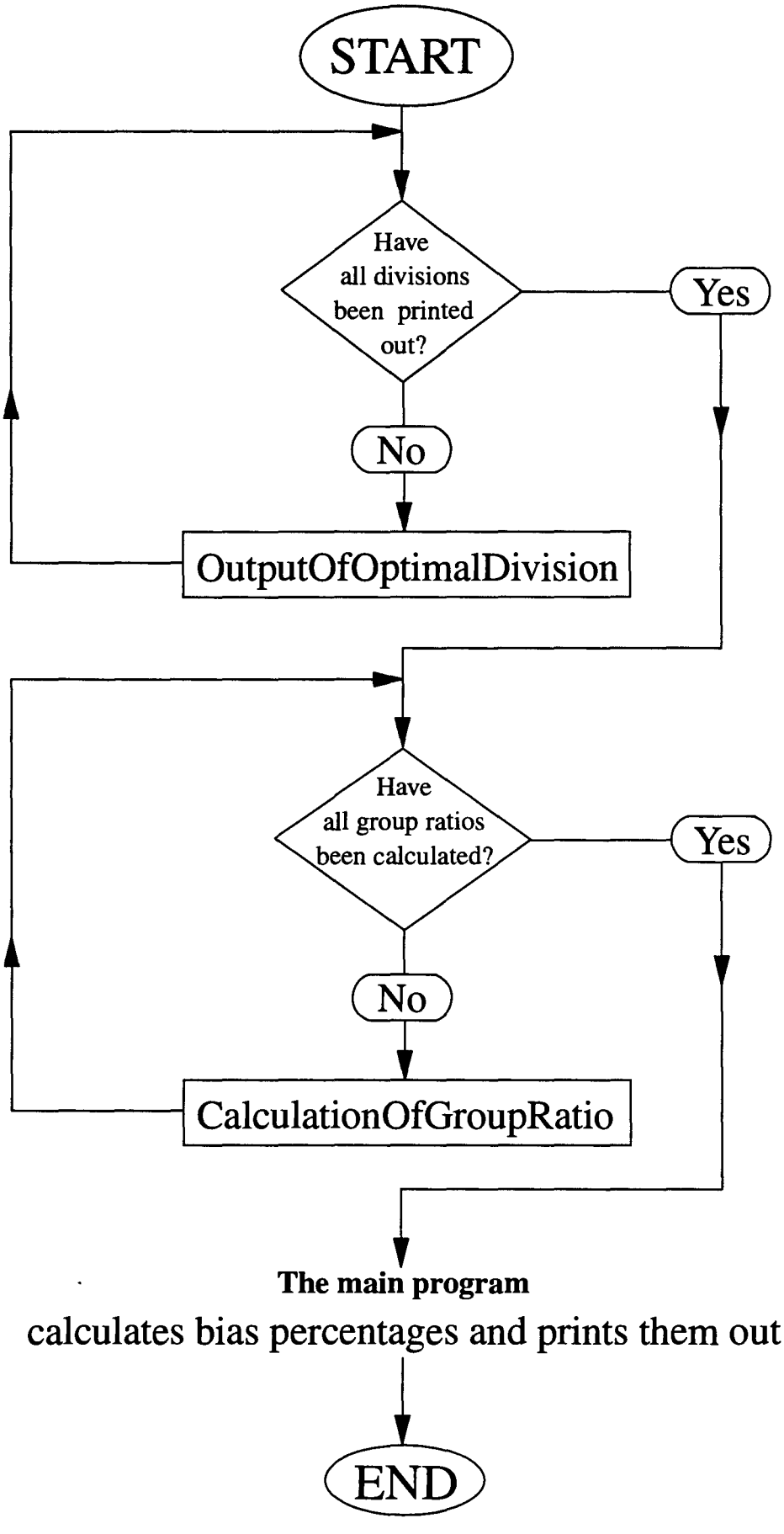


Division section



Flow charts of the procedures ClusterDivision and QuotaDivision can be found on pages A 60 and A 61 respectively.

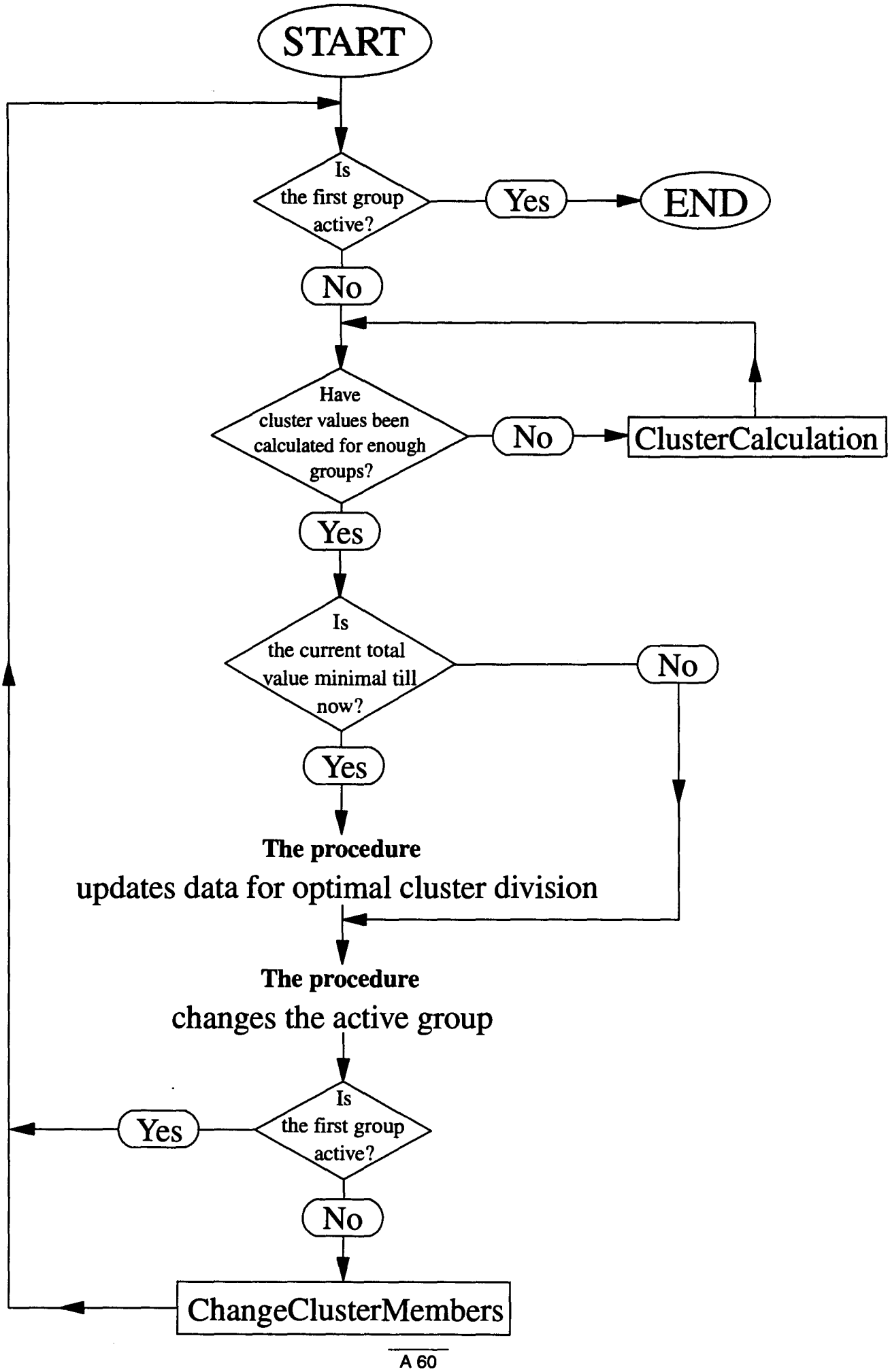
Bias calculation section



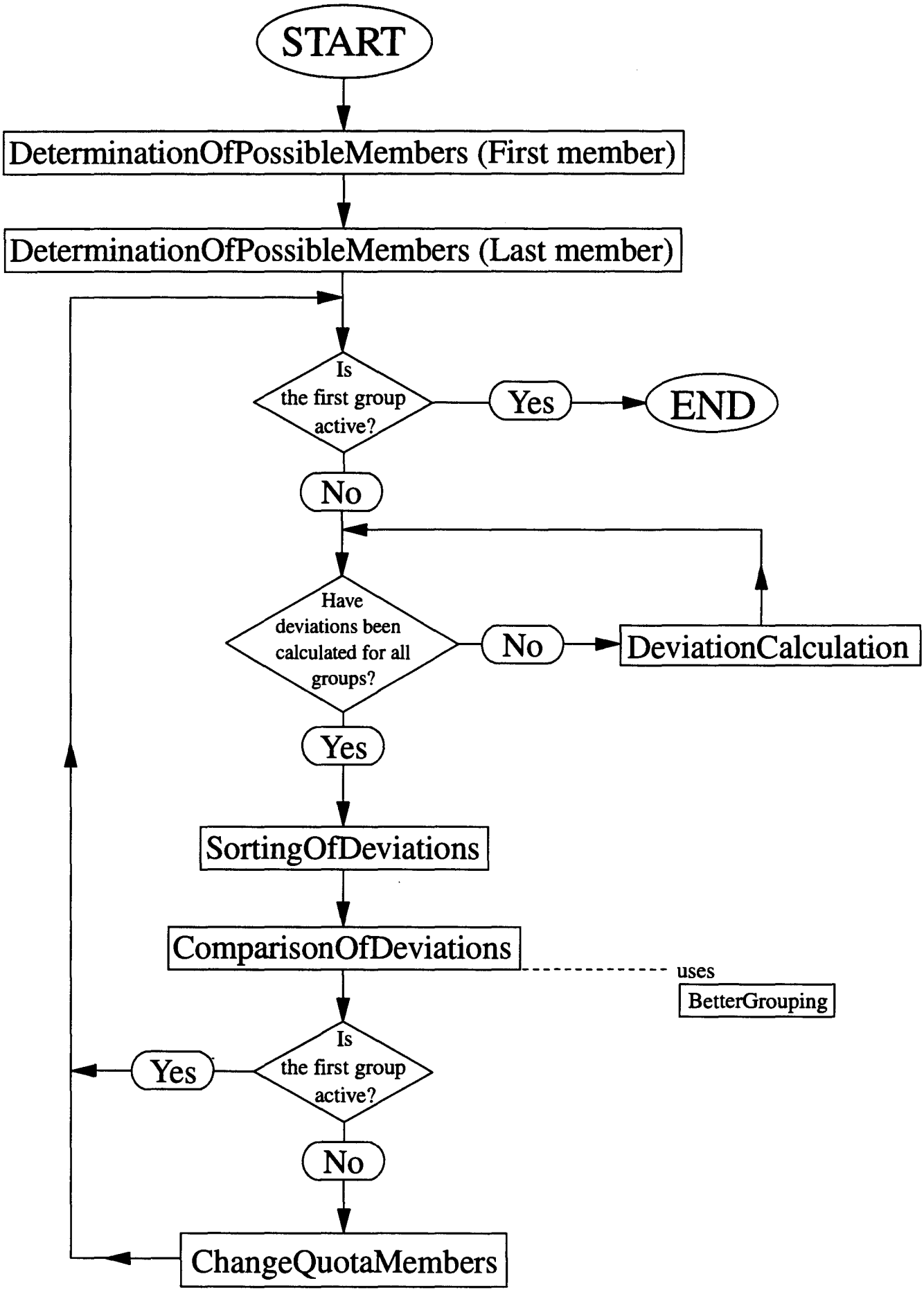
The main program

calculates bias percentages and prints them out

The ClusterDivision procedure



The QuotaDivision procedure



```
PROGRAM MatrixBias (Ice63, TwoDimLFGamsOutput, BiasApport, Bias);

{Constants, types and variables which are used in most sections are declared
 right below, while the ones which are used in one or two sections are
 declared for the section they first appear.}

{Second program in the command sequence COMPLETERUN.COM}

CONST NumberOfConstituencies =   8;
      NumberOfParties =          4;
      NumberOfSeats =           60;
      WordLength =               3;
      NumberOfMatrixMethods =    5;

VAR i:    integer; {ConstituencyNumber}
    j:    integer; {PartyNumber}
    p:    integer; {MatrixMethodNumber}
    l:    integer; {CellNumber}
    Bias: text;

{------------------------------------------------------------------------------}

{Section for reading of data}

TYPE String =              PACKED ARRAY [1..WordLength] OF char;
     Constituencies =      ARRAY [1..NumberOfConstituencies] OF String;
     Parties =             ARRAY [1..NumberOfParties] OF String;
     Methods =             ARRAY [1..NumberOfMatrixMethods] OF String;
     ConstituencyVector =  ARRAY [1..NumberOfConstituencies] OF integer;
     PartyVector =         ARRAY [1..NumberOfParties] OF integer;
     IntegerMatrix =       ARRAY [1..NumberOfConstituencies, 1..NumberOfParties]
                           OF integer;
     TripleMatrix =        ARRAY [1..NumberOfConstituencies, 1..NumberOfParties,
                           1..NumberOfMatrixMethods] OF integer;

VAR SumVotes:                    integer;
    SumWholeSeats:               integer;
    SumFractionalSeats:          integer;
    Character:                   char;
    Optimal:                     boolean;
    Ice63:                       text;
    TwoDimLFGamsOutput:          text;
    BiasApport:                  text;
    ConstituencyName:            Constituencies;
    PartyName:                   Parties;
    MethodName:                  Methods;
    ConstituencyVotes:           ConstituencyVector;
    WholeConstituencySeats:      ConstituencyVector;
    FractionalConstituencySeats: ConstituencyVector;
    ConstituencyRepresentatives: ConstituencyVector;
    PartyVotes:                  PartyVector;
    WholePartySeats:             PartyVector;
    FractionalPartySeats:        PartyVector;
    PartyRepresentatives:        PartyVector;
    Votes:                       IntegerMatrix;
    FractionalSeats:             IntegerMatrix;
    WholeSeats:                  IntegerMatrix;
    Representatives:             IntegerMatrix;
    LFApportionment:             IntegerMatrix;
    MatrixApportionments:        TripleMatrix;

{------------------------------------------------------------------------------}

{Section for finding the fair share allocation.}

CONST AllowedDiscrepancy = 1.0E-4;

TYPE RealConstituencyVector = ARRAY [1..NumberOfConstituencies] OF double;
     RealPartyVector =        ARRAY [1..NumberOfParties] OF double;
     RealMatrix =             ARRAY [1..NumberOfConstituencies,
                                     1..NumberOfParties] OF double;

VAR DoubleIterations:                integer;
    Discrepancy:                     double;
    CurrentFairShare:                double;
    ContinuousConstituencyMultiplie: RealConstituencyVector;
    FairShareConstituencySum:        RealConstituencyVector;
    ContinuousPartyMultiplier:       RealPartyVector;
    FairSharePartySum:               RealPartyVector;
    InitialFairShares:               RealMatrix;
    FinalFairShares:                 RealMatrix;

{------------------------------------------------------------------------------}
```

```
{Section for calculation of seat differences}

TYPE TriangleMatrix = ARRAY [1..NumberOfMatrixMethods,
                             1..NumberOfMatrixMethods] OF integer;

VAR SecondMethod:   integer;
    SeatDifference: TriangleMatrix;

{------------------------------------------------------------------------------}

{Section for cells}

CONST DividingFactor = 1;

TYPE BooleanPartyVector = ARRAY [1..NumberOfParties] OF boolean;
     SevenString =        PACKED ARRAY [1..7] OF char;
     CellData =           RECORD Name:               SevenString;
                                 FairShare:          double;
                                 ConstituencyNumber: integer;
                                 PartyNumber:        integer;
                                 END; {CellData)
     CellRecordVector =   ARRAY [1..(NumberOfConstituencies * NumberOfParties)]
                                 OF CellData;

VAR f:                        integer;
    NumberOfPartyDivisions:   integer;
    NumberOfSelectedParties:  integer;
    NumberOfSelectedSeats:    integer;
    NumberOfCellObjects:      integer;
    TemporaryFile:            text;
    SelectedParty:            BooleanPartyVector;
    Cell:                     CellRecordVector;

{------------------------------------------------------------------------------}

{Section for divisions}

CONST MaxNumberOfGroups = 4;
      NumberOfDivisions = 4;

TYPE Divisions =        ARRAY [1..NumberOfDivisions] OF SevenString;
     GroupVector =      ARRAY [1..MaxNumberOfGroups] OF integer;
     RealGroupVector =  ARRAY [1..MaxNumberOfGroups] OF double;
     RealMethodVector = ARRAY [1..NumberOfMatrixMethods] OF double;
     DivisionMatrix =   ARRAY [1..MaxNumberOfGroups, 1..NumberOfDivisions]
                              OF integer;

VAR d:                              integer; {DivisionNumber)
    g:                              integer; {GroupNumber)
    SecondGroup:                    integer;
    NumberOfGroups:                 integer;
    ActiveGroup:                    integer;
    VisitedCombinations:            integer;
    ClusterCombinations:            integer;
    QuotaCombinations:              integer;
    MinimalValue:                   double;
    ClusterValue:                   double;
    AverageFairShare:               double;
    MaxDeviation:                   double;
    GroupFairShare:                 double;
    GroupDeviation:                 double;
    EmptySizeDivision:              boolean;
    DivisionName:                   Divisions;
    FirstGroupMember:               GroupVector;
    LastGroupMember:                GroupVector;
    FirstPossibleFirstGroupMember:  GroupVector;
    LastPossibleFirstGroupMember:   GroupVector;
    Deviations:                     RealGroupVector;
    BestDeviations:                 RealGroupVector;
    GroupRatio:                     RealMethodVector;
    OptimalFirstGroupMember:        DivisionMatrix;
    OptimalLastGroupMember:         DivisionMatrix;

{------------------------------------------------------------------------------}

{Section for bias calculation}

TYPE GroupRatioMatrix = ARRAY [1..MaxNumberOfGroups, 1..NumberOfDivisions,
                               1..NumberOfMatrixMethods] OF double;
     BiasMatrix =       ARRAY [1..MaxNumberOfGroups, 1..MaxNumberOfGroups,
                               1..NumberOfDivisions, 1..NumberOfMatrixMethods]
                               OF double;

VAR GroupRatios:    GroupRatioMatrix;
    BiasPercentage: BiasMatrix;

{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}
```

```
PROCEDURE ReadingOfElectionData (VAR ElectionData: text);

BEGIN {ReadingOfElectionData}

reset (ElectionData);

FOR i:= 1 TO NumberOfConstituencies DO
   readln (ElectionData, ConstituencyName [i]);
readln (ElectionData);

FOR j:= 1 TO NumberOfParties DO
   readln (ElectionData, PartyName [j]);
readln (ElectionData);

FOR i:= 1 TO NumberOfConstituencies DO
   FOR j:= 1 TO NumberOfParties DO
      read (ElectionData, Votes [i,j]);

END; {ReadingOfElectionData}

{-------------------------------------------------------------------------------}
{-------------------------------------------------------------------------------}

PROCEDURE CalculationOfSums (Matrix:              IntegerMatrix;
                             VAR ConstituencySum: ConstituencyVector;
                             VAR PartySum:        PartyVector;
                             VAR Sum:             integer);

BEGIN {CalculationOfSums}

FOR i:= 1 TO NumberOfConstituencies DO
   ConstituencySum [i] := 0;

FOR j:= 1 To NumberOfParties DO
   PartySum [j] := 0;

Sum:= 0;

FOR i:= 1 TO NumberOfConstituencies DO
   FOR j:= 1 TO NumberOfParties DO
      ConstituencySum [i] := ConstituencySum [i] + Matrix [i,j];

FOR j:= 1 TO NumberOfParties DO
   FOR i:= 1 TO NumberOfConstituencies DO
      PartySum [j] := PartySum [j] + Matrix [i,j];

FOR i:= 1 TO NumberOfConstituencies DO
   FOR j:= 1 TO NumberOfParties DO
      Sum:= Sum + Matrix [i,j];

END; {CalculationOfSums}

{-------------------------------------------------------------------------------}
{-------------------------------------------------------------------------------}

PROCEDURE WritingOfLine (VAR OutputFile: text);

BEGIN {WritingOfLine}

writeln (OutputFile, '-------------------------------------------------------',
   '------------------------');

END; {WritingOfLine}

{-------------------------------------------------------------------------------}
{-------------------------------------------------------------------------------}
```

```
PROCEDURE ReadingOfGamsData (VAR GamsData: text);

{Reads the ouput data from a gams outputfile, i.e. a file with an original
 extension of .lis.}

CONST SeekLength = 11;

TYPE LongString =   PACKED ARRAY [1..SeekLength] OF char;
     StringMatrix = ARRAY [1..NumberOfConstituencies,
                           1..NumberOfParties] OF String;

VAR DataIndicator:              LongString;
    SeekString:                 LongString;
    TestWord:                   LongString;
    Redundant:                  LongString;
    Status:                     LongString;
    MiniRedundant:              String;
    Level:                      StringMatrix;

BEGIN {ReadingOfGamsData}

reset (GamsData);

SeekString:= '**** SOLVER';
REPEAT
   readln (GamsData, TestWord)
UNTIL (TestWord = SeekString);

read (GamsData, DataIndicator);
IF (DataIndicator = '**** MODEL ') THEN
read (GamsData, Redundant);
read (GamsData, MiniRedundant);
read (GamsData, Status);
IF (Status = 'OPTIMAL    ') THEN
   Optimal:= True;

SeekString:= '---- VAR A ';
REPEAT
   readln (GamsData, TestWord)
UNTIL (TestWord = SeekString);

FOR i:= 1 TO NumberOfConstituencies DO BEGIN
   j:= 1;
   WHILE j <= NumberOfParties DO BEGIN
      read (GamsData, Redundant);
      read (GamsData, DataIndicator);
      IF (DataIndicator = '  .         ') THEN BEGIN
         read (GamsData, Level [i,j]);
         j:= j + 1;
      END; {Then}
      readln (GamsData);
   END; {While}
END; {For i}

FOR i:= 1 TO NumberOfConstituencies DO
   FOR j:= 1 TO NumberOfParties DO BEGIN
      IF Level [i,j] = '1.0' THEN
         FractionalSeats [i,j] := 1;
      IF Level [i,j] = ' . ' THEN
         FractionalSeats [i,j] := 0;
   END; {For j}

END; {ReadingOfGamsData}

{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}
```

```
PROCEDURE ReadingOfRepresentativesData (VAR RepresentativesData: text);

{Reads ouput data from a file containing a representative matrix}

CONST SeekLength = 26;

TYPE LongString = PACKED ARRAY [1..SeekLength] OF char;

VAR Redundant:       String;
    MethodIndicator: String;
    SeekString:      LongString;
    TestWord:        LongString;

BEGIN {ReadingOfRepresentativesData}

reset (RepresentativesData);

SeekString:= 'Matrix apportionment with ';

REPEAT
   read (RepresentativesData, TestWord);
   readln (RepresentativesData, MethodIndicator)
UNTIL (TestWord = SeekString) AND (MethodIndicator = MethodName [p]);

readln (RepresentativesData);
readln (RepresentativesData);
FOR i:= 1 TO NumberOfConstituencies DO BEGIN
   read (RepresentativesData, Redundant);
   FOR j:= 1 TO NumberOfParties DO
      read (RepresentativesData, MatrixApportionments [i,j,p]);
   readln (RepresentativesData);
END; {For i}

END; {ReadingOfRepresentativesData}

{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}
```

```
PROCEDURE OutputOfIntegerMatrices (VAR OutputFile: text;
                                   FieldWidth:     integer;
                                   SumFieldWidth:  integer;
                                   Matrix:         IntegerMatrix);

VAR s:                integer;
    t:                integer;
    RemainingParties: integer;
    ColumnsPerLine:   integer;
    NumberOfTables:   integer;
    Sum:              integer;
    ConstituencySum:  ConstituencyVector;
    PartySum:         PartyVector;

BEGIN {OutputOfIntegerMatrices}

CalculationOfSums (Matrix, ConstituencySum, PartySum, Sum);

RemainingParties:= NumberOfParties;
ColumnsPerLine:= ((77 - SumFieldWidth) DIV FieldWidth) + 1;
IF (NumberOfParties + 1) MOD ColumnsPerLine = 0 THEN
   NumberOfTables:= ((NumberOfParties + 1) DIV ColumnsPerLine)
ELSE NumberOfTables:= ((NumberOfParties + 1) DIV ColumnsPerLine) + 1;

writeln (OutputFile);
FOR t:= 1 TO NumberOfTables DO BEGIN
   write (OutputFile, '    ');
   FOR j:= (NumberOfParties - RemainingParties + 1) TO MIN(NumberOfParties,
           (NumberOfParties - RemainingParties + ColumnsPerLine)) DO
      write (OutputFile, PartyName [j] :FieldWidth);
   IF t = NumberOfTables THEN BEGIN
      FOR s:= 1 TO (SumFieldWidth - 3) DO
         write (OutputFile, ' ');
      writeln (OutputFile, 'Sum');
   END {Then}
   ELSE writeln (OutputFile);

   FOR i:= 1 TO NumberOfConstituencies DO BEGIN
      write (OutputFile, ConstituencyName [i]);
      FOR j:= (NumberOfParties - RemainingParties + 1) TO MIN(NumberOfParties,
              (NumberOfParties - RemainingParties + ColumnsPerLine)) DO
         write (OutputFile, Matrix [i,j] :FieldWidth);
      IF t = NumberOfTables THEN
         writeln (OutputFile, ConstituencySum [i] :SumFieldWidth)
      ELSE writeln (OutputFile);
   END; {For i}
   writeln (OutputFile);

   write (OutputFile, 'Sum');
   FOR j:= (NumberOfParties - RemainingParties + 1) TO MIN(NumberOfParties,
           (NumberOfParties - RemainingParties + ColumnsPerLine)) DO
      write (OutputFile, PartySum [j] :FieldWidth);
   IF t = NumberOfTables THEN
      writeln (OutputFile, Sum:SumFieldWidth)
   ELSE BEGIN
      writeln (OutputFile);
      writeln (OutputFile);
   END; {Else}
   RemainingParties:= RemainingParties - ColumnsPerLine;
   writeln (OutputFile);
END; {For t}

END; {OutputOfIntegerMatrices}

{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}
```

```
PROCEDURE CalculationOfRealSums (Matrix:              RealMatrix;
                                 VAR ConstituencySum: RealConstituencyVector;
                                 VAR PartySum:        RealPartyVector;
                                 VAR Sum:             double);

BEGIN {CalculationOfRealSums}

FOR i:= 1 TO NumberOfConstituencies DO
   ConstituencySum [i] := 0;

FOR j:= 1 To NumberOfParties DO
   PartySum [j] := 0;

Sum:= 0;

FOR i:= 1 TO NumberOfConstituencies DO
   FOR j:= 1 TO NumberOfParties DO
      ConstituencySum [i] := ConstituencySum [i] + Matrix [i,j];

FOR j:= 1 TO NumberOfParties DO
   FOR i:= 1 TO NumberOfConstituencies DO
      PartySum [j] := PartySum [j] + Matrix [i,j];

FOR i:= 1 TO NumberOfConstituencies DO
   FOR j:= 1 TO NumberOfParties DO
      Sum:= Sum + Matrix [i,j];

END; {CalculationOfRealSums}

{-------------------------------------------------------------------------------}
{-------------------------------------------------------------------------------}

PROCEDURE OutputOfRealMatrices (VAR OutputFile: text;
                                FieldWidth:     integer;
                                SumFieldWidth:  integer;
                                DecimalPlaces:  integer;
                                Matrix:         RealMatrix);

VAR s:                integer;
    t:                integer;
    RemainingParties: integer;
    ColumnsPerLine:   integer;
    NumberOfTables:   integer;
    Sum:              double;
    ConstituencySum:  RealConstituencyVector;
    PartySum:         RealPartyVector;

BEGIN {OutputOfRealMatrices}

CalculationOfRealSums (Matrix, ConstituencySum, PartySum, Sum);

RemainingParties:= NumberOfParties;
ColumnsPerLine:= ((77 - SumFieldWidth) DIV FieldWidth) + 1;
IF (NumberOfParties + 1) MOD ColumnsPerLine = 0 THEN
   NumberOfTables:= ((NumberOfParties + 1) DIV ColumnsPerLine)
ELSE NumberOfTables:= ((NumberOfParties + 1) DIV ColumnsPerLine) + 1;

writeln (OutputFile);
FOR t:= 1 TO NumberOfTables DO BEGIN
   write (OutputFile, '   ');
   FOR j:= (NumberOfParties - RemainingParties + 1) TO MIN(NumberOfParties,
            (NumberOfParties - RemainingParties + ColumnsPerLine)) DO
      write (OutputFile, PartyName [j] :FieldWidth);
   IF t = NumberOfTables THEN BEGIN
      FOR s:= 1 TO (SumFieldWidth - 3) DO
         write (OutputFile, ' ');
      writeln (OutputFile, 'Sum');
   END {Then}
   ELSE writeln (OutputFile);
```

```
   FOR i:= 1 TO NumberOfConstituencies DO BEGIN
      write (OutputFile, ConstituencyName [i]);
      FOR j:= (NumberOfParties - RemainingParties + 1) TO MIN(NumberOfParties,
               (NumberOfParties - RemainingParties + ColumnsPerLine)) DO
         write (OutputFile, Matrix [i,j] :FieldWidth:DecimalPlaces);
      IF t = NumberOfTables THEN
         writeln (OutputFile, ConstituencySum [i] :SumFieldWidth:DecimalPlaces)
      ELSE writeln (OutputFile);
   END; {For i}
   writeln (OutputFile);

   write (OutputFile, 'Sum');
   FOR j:= (NumberOfParties - RemainingParties + 1) TO MIN(NumberOfParties,
            (NumberOfParties - RemainingParties + ColumnsPerLine)) DO
      write (OutputFile, PartySum [j] :FieldWidth:DecimalPlaces);
   IF t = NumberOfTables THEN
      writeln (OutputFile, Sum:SumFieldWidth:DecimalPlaces)
   ELSE BEGIN
      writeln (OutputFile);
      writeln (OutputFile);
   END; {Else}
   RemainingParties:= RemainingParties - ColumnsPerLine;
   writeln (OutputFile);
END; {For t}

END; {OutputOfRealMatrices}

{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}

PROCEDURE ScalingOfContinuousMultipliers;

BEGIN {ScalingOfContinuousMultipliers}

FOR i:= 1 TO NumberOfConstituencies DO BEGIN
   FairShareConstituencySum [i] := 0;
   FOR j:= 1 TO NumberOfParties DO BEGIN
      CurrentFairShare:= (InitialFairShares [i,j] *
         ContinuousConstituencyMultiplie [i] * ContinuousPartyMultiplier [j]);
      FairShareConstituencySum [i] :=
         FairShareConstituencySum [i] + CurrentFairShare;
   END; {For j}
   Discrepancy:= Discrepancy +
      ABS(FairShareConstituencySum [i] - ConstituencyRepresentatives [i]);
   ContinuousConstituencyMultiplie [i] :=
      ContinuousConstituencyMultiplie [i] *
     (ConstituencyRepresentatives [i] / FairShareConstituencySum [i]);
END; {For i}

FOR j:= 1 TO NumberOfParties DO BEGIN
   FairSharePartySum [j] := 0;
   FOR i:= 1 TO NumberOfConstituencies DO BEGIN
      CurrentFairShare:= (InitialFairShares [i,j] *
         ContinuousConstituencyMultiplie [i] * ContinuousPartyMultiplier [j]);
      FairSharePartySum [j] := FairSharePartySum [j] + CurrentFairShare;
   END; {For i}
   Discrepancy:= Discrepancy +
      ABS(FairSharePartySum [j] - PartyRepresentatives [j]);
   IF FairSharePartySum [j] = 0 THEN
      ContinuousPartyMultiplier [j] := 0
   ELSE ContinuousPartyMultiplier [j] := ContinuousPartyMultiplier [j] *
       (PartyRepresentatives [j] / FairSharePartySum [j]);
END; {For j}

END; {ScalingOfContinuousMultipliers}

{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}
```

```
PROCEDURE SortingOfCells (Start:  integer;
                          Finish: integer);

{Sorts the cells by quicksort, see page 533.}

VAR Left:       integer;
    Right:      integer;
    StartValue: double;
    Temporary:  CellData;

BEGIN {SortingOfCells}

Left:= Start;
Right:= Finish;
StartValue:= Cell [(Start + Finish) DIV 2].FairShare;

REPEAT
   WHILE Cell [Left].FairShare > StartValue DO
      Left:= Left + 1;
   WHILE Cell [Right].FairShare < StartValue DO
      Right:= Right - 1;
   IF Left <= Right THEN BEGIN
      Temporary:= Cell [Left];
      Cell [Left]:= Cell [Right];
      Cell [Right]:= Temporary;
      Left:= Left + 1;
      Right:= Right - 1;
   END; {Then}
UNTIL Right <= Left;

IF Start < Right THEN
   SortingOfCells (Start, Right);
IF Left < Finish THEN
   SortingOfCells (Left, Finish);

END; {SortingOfCells}

{-------------------------------------------------------------------------------}
{-------------------------------------------------------------------------------}
{-------------------------------------------------------------------------------}

PROCEDURE NumberDivision;

VAR Number:  integer;
    Surplus: integer;

BEGIN {NumberDivision}

Number:= NumberOfCellObjects DIV NumberOfGroups;
Surplus:= NumberOfCellObjects MOD NumberOfGroups;

g:= 1;
WHILE (g <= (NumberOfGroups - Surplus)) AND (g < NumberOfGroups) DO BEGIN
   OptimalLastGroupMember [g,d] := OptimalFirstGroupMember [g,d] + Number - 1;
   OptimalFirstGroupMember [(g + 1), d] := OptimalLastGroupMember [g,d] + 1;
   g:= g + 1;
END; {While g}

WHILE g < NumberOfGroups DO BEGIN
   OptimalLastGroupMember [g,d] := OptimalFirstGroupMember [g,d] + Number;
   OptimalFirstGroupMember [(g + 1), d] := OptimalLastGroupMember [g,d] + 1;
   g:= g + 1;
END; {While g}

END; {NumberDivision}

{-------------------------------------------------------------------------------}
{-------------------------------------------------------------------------------}
```

```
PROCEDURE SizeDivision;

VAR MaxSize:      double;
    IntervalSize: double;

BEGIN {SizeDivision}

MaxSize:= Cell [1].FairShare;
IntervalSize:= MaxSize / NumberOfGroups;

l:= 1;
g:= 1;
WHILE g < NumberOfGroups DO BEGIN
   WHILE (Cell [l].FairShare > MaxSize - (g * IntervalSize)) AND
      (l <= NumberOfCellObjects) DO
      l:= l + 1;
   OptimalLastGroupMember [g,d] := l - 1;
   OptimalFirstGroupMember [(g + 1),d] := l;
   g:= g + 1;
   l:= l - 1;
END; {While g}

FOR g:= 1 TO NumberOfGroups DO
   IF OptimalLastGroupMember [g,d] < OptimalFirstGroupMember [g,d] THEN BEGIN
      writeln (Bias, 'No size division with this number of groups ');
      EmptySizeDivision:= True;
   END; {Then}

END; {SizeDivision}

{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}

PROCEDURE ClusterCalculation (FirstMember: integer;
                              LastMember:  integer);

{Calculates D(C) for the cluster defined by FirstMember and LastMember.}

VAR ClusterAverage: double;
    ClusterSum:     double;
    Variance:       double;

BEGIN {ClusterCalculation}

ClusterSum:= 0;
ClusterValue:= 0;
FOR l:= FirstMember TO LastMember DO
   ClusterSum:= ClusterSum + Cell [l].FairShare;
ClusterAverage:= ClusterSum / (LastMember - FirstMember + 1);

FOR l:= FirstMember TO LastMember DO BEGIN
   Variance:= SQR(Cell [l].FairShare - ClusterAverage);
   ClusterValue:= ClusterValue + Variance;
END; {For l}

END; {ClusterCalculation}

{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}

PROCEDURE ChangeClusterMembers;

{This procedure is used for the second to the penultimate cluster.}

BEGIN {ChangeClusterMembers}

FirstGroupMember [ActiveGroup] := FirstGroupMember [ActiveGroup] + 1;
LastGroupMember [ActiveGroup - 1]:= FirstGroupMember [ActiveGroup] - 1;
FOR g:= (ActiveGroup + 1) TO NumberOfGroups DO
   FirstGroupMember [g] := FirstGroupMember [g - 1] + 1;
FOR g:= ActiveGroup TO (NumberOfGroups - 1) DO
   LastGroupMember [g] := FirstGroupMember [g];
LastGroupMember [NumberOfGroups] := NumberOfCellObjects;

END; {ChangeClusterMembers}

{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}
```

```
PROCEDURE ClusterDivision;

VAR Value: double;

BEGIN {ClusterDivision}

FOR g:= 1 TO NumberOfGroups DO BEGIN
   FirstGroupMember [g] := g;
   LastGroupMember [g] := FirstGroupMember [g];
END; {For g}
LastGroupMember [NumberOfGroups] := NumberOfCellObjects;

MinimalValue:= 1.0E+38;
VisitedCombinations:= 0;
ActiveGroup:= NumberOfGroups;

WHILE ActiveGroup > 1 DO BEGIN

   VisitedCombinations:= VisitedCombinations + 1;
   Value:= 0;
   g:= 1;

   WHILE (Value <= MinimalValue) AND (g <= NumberOfGroups) DO BEGIN
      ClusterCalculation (FirstGroupMember [g], LastGroupMember [g]);
      Value:= Value + ClusterValue;
      g:= g + 1;
   END; {While}

   IF Value < MinimalValue THEN BEGIN
      MinimalValue:= Value;
      FOR g:= 1 TO NumberOfGroups DO BEGIN
         OptimalFirstGroupMember [g,d] := FirstGroupMember [g];
         OptimalLastGroupMember [g,d] := LastGroupMember [g];
      END; {For g}
   END; {Then}

   WHILE (FirstGroupMember [ActiveGroup] =
      NumberOfCellObjects - NumberOfGroups + ActiveGroup) AND
     (ActiveGroup > 1) DO
      ActiveGroup:= ActiveGroup - 1;

   IF ActiveGroup > 1 THEN BEGIN
      ChangeClusterMembers;
      ActiveGroup:= NumberOfGroups;
   END; {Then}

END; {While}

ClusterCombinations:= 1;
FOR g:= 1 TO (NumberOfGroups - 1) DO
   ClusterCombinations:= ClusterCombinations * (NumberOfCellObjects - g);
FOR g:= 1 TO (NumberOfGroups - 1) DO
   ClusterCombinations:= ClusterCombinations DIV (NumberOfGroups - g);

IF ClusterCombinations <> VisitedCombinations THEN BEGIN
   write (Bias, 'Something is wrong with the enumeration of ');
   writeln (Bias, 'cluster combinations ');
END; {Then}

END; {ClusterDivision}

{-------------------------------------------------------------------------}
{-------------------------------------------------------------------------}

PROCEDURE DeterminationOfPossibleMembers
                                  (Deviation:                    double;
                                   VAR PossibleFirstGroupMember: GroupVector);

BEGIN {DeterminationOfPossibleMembers}

FOR g:= 2 TO NumberOfGroups DO BEGIN
   l:= PossibleFirstGroupMember [g - 1];
   GroupFairShare:= 0;
   WHILE (GroupFairShare < (AverageFairShare + Deviation)) AND
      (l < NumberOfCellObjects) DO BEGIN
      GroupFairShare:= GroupFairShare + Cell [l].FairShare;
      l:= l + 1;
   END; {While}
   PossibleFirstGroupMember [g] :=
      MAX(l, MIN((PossibleFirstGroupMember [g - 1] + 1), NumberOfCellObjects));
END; {For g}

END; {DeterminationOfPossibleMembers}

{-------------------------------------------------------------------------}
{-------------------------------------------------------------------------}
```

```
PROCEDURE DeviationCalculation (FirstMember: integer;
                                LastMember:  integer);

{Calculates the absolute deviation from the average quota size for the group
 defined by FirstMember and LastMember.}

BEGIN {DeviationCalculation}

GroupFairShare:= 0;
FOR l:= FirstMember TO LastMember DO
   GroupFairShare:= GroupFairShare + Cell [l].FairShare;
GroupDeviation:= ABS(GroupFairShare - AverageFairShare);

END; {DeviationCalculation}

{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}

PROCEDURE SortingOfDeviations (Start:  integer;
                               Finish: integer);

{Sorts the deviations by quicksort.}

VAR Left:       integer;
    Right:      integer;
    StartValue: double;
    Temporary:  double;

BEGIN {SortingOfDeviations}

Left:= Start;
Right:= Finish;
StartValue:= Deviations [(Start + Finish) DIV 2];

REPEAT
   WHILE Deviations [Left] > StartValue DO
      Left:= Left + 1;
   WHILE Deviations [Right] < StartValue DO
      Right:= Right - 1;
   IF Left <= Right THEN BEGIN
      Temporary:= Deviations [Left];
      Deviations [Left] := Deviations [Right];
      Deviations [Right] := Temporary;
      Left:= Left + 1;
      Right:= Right - 1;
   END; {Then}
UNTIL Right <= Left;

IF Start < Right THEN
   SortingOfDeviations (Start, Right);
IF Left < Finish THEN
   SortingOfDeviations (Left, Finish);

END; {SortingOfDeviations}

{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}

PROCEDURE BetterGrouping;

BEGIN {BetterGrouping}

BestDeviations:= Deviations;
MaxDeviation:= BestDeviations [1];
FOR g:= 1 TO NumberOfGroups DO BEGIN
   OptimalFirstGroupMember [g,d] := FirstGroupMember [g];
   OptimalLastGroupMember [g,d] := LastGroupMember [g];
END; {For g}

END; {BetterGrouping}

{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}

PROCEDURE ComparisonOfDeviations;

BEGIN {ComparisonOfDeviations}

IF g <= NumberOfGroups THEN
   IF Deviations [g] < BestDeviations [g] THEN
      BetterGrouping
   ELSE IF Deviations [g] = BestDeviations [g] THEN BEGIN
           g:= g + 1;
           ComparisonOfDeviations;
        END; {Then}

END; {ComparisonOfDeviations}

{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}
```

```
PROCEDURE ChangeQuotaMembers;

BEGIN {ChangeQuotaMembers}

FirstGroupMember [ActiveGroup] := FirstGroupMember [ActiveGroup] + 1;
LastGroupMember [ActiveGroup - 1] := FirstGroupMember [ActiveGroup] - 1;
FOR g:= (ActiveGroup + 1) TO NumberOfGroups DO BEGIN
   FirstGroupMember [g] := FirstPossibleFirstGroupMember [g];
   LastGroupMember [g - 1] := FirstGroupMember [g] - 1;
END; {For g}

END; {ChangeQuotaMembers}

{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}

PROCEDURE QuotaDivision (SeatTotal: integer);

BEGIN {QuotaDivision}

AverageFairShare:= SeatTotal / NumberOfGroups;
l:= 1;
GroupFairShare:= 0;
WHILE GroupFairShare < AverageFairShare DO BEGIN
   GroupFairShare:= GroupFairShare + Cell [l].FairShare;
   l:= l + 1;
END; {While}
MaxDeviation:= Cell [(l - 1)].FairShare;

FOR g:= 1 TO NumberOfGroups DO
   BestDeviations [g] := MaxDeviation;

FirstPossibleFirstGroupMember [1] := 1;
LastPossibleFirstGroupMember [1] := 1;

DeterminationOfPossibleMembers (-MaxDeviation, FirstPossibleFirstGroupMember);
DeterminationOfPossibleMembers ((MaxDeviation + 1.0E-16),
                                LastPossibleFirstGroupMember);

FOR g:= 1 TO NumberOfGroups DO
   FirstGroupMember [g] := FirstPossibleFirstGroupMember [g];

FOR g:= 1 TO (NumberOfGroups - 1) DO
   LastGroupMember [g] := FirstGroupMember [g + 1] - 1;
LastGroupMember [NumberOfGroups] := NumberOfCellObjects;

VisitedCombinations:= 0;
ActiveGroup:= NumberOfGroups;

WHILE ActiveGroup > 1 DO BEGIN
   VisitedCombinations:= VisitedCombinations + 1;

   g:= 1;
   GroupDeviation:= 0;
   WHILE (g <= NumberOfGroups) AND (GroupDeviation <= MaxDeviation) DO BEGIN
      DeviationCalculation (FirstGroupMember [g], LastGroupMember [g]);
      Deviations [g] := GroupDeviation;
      g:= g + 1;
   END; {While}

   IF g = (NumberOfGroups + 1) THEN BEGIN
      SortingOfDeviations (1,NumberOfGroups);
      g:= 1;
      ComparisonOfDeviations;
   END; {Then}

   WHILE (ActiveGroup > 1) AND (FirstGroupMember [ActiveGroup] =
      LastPossibleFirstGroupMember [ActiveGroup]) DO
      ActiveGroup:= ActiveGroup - 1;

   IF ActiveGroup > 1 THEN BEGIN
      ChangeQuotaMembers;
      ActiveGroup:= NumberOfGroups;
   END; {Then}

END; {While}

QuotaCombinations:= 1;
FOR g:= 1 TO NumberOfGroups DO
   QuotaCombinations:= QuotaCombinations * (LastPossibleFirstGroupMember [g] -
      FirstPossibleFirstGroupMember [g] + 1);

IF QuotaCombinations <> VisitedCombinations THEN BEGIN
   write (Bias, 'Something is wrong with the investigation of possible ');
   writeln (Bias, 'quota combinations ');
END; {Then}

END; {QuotaDivision}

{------------------------------------------------------------------------------}
{------------------------------------------------------------------------------}
```

```
PROCEDURE OutputOfOptimalDivision;

BEGIN {OutputOfOptimalDivision}

CASE d OF
   1, 2: ;
   3: BEGIN
         writeln (Bias, 'Minimal value          ', MinimalValue:6:2);
         writeln (Bias, 'Number of combinations ', ClusterCombinations:3);
         writeln (Bias);
      END;
   4: BEGIN
         writeln (Bias, 'Largest deviation      ', MaxDeviation:6:2);
         writeln (Bias, 'Number of combinations ', QuotaCombinations:3);
         writeln (Bias);
      END;
END; {Case}

writeln (Bias, 'The ', DivisionName [d], ' division  is ');
IF (d = 2) AND EmptySizeDivision THEN BEGIN
   writeln (Bias);
   writeln (Bias, 'Empty size division with ', NumberOfGroups:1, ' groups')
END {Then}
ELSE
FOR g:= 1 TO NumberOfGroups DO BEGIN
   write (Bias, 'Group ', g:1, ' includes cellobject ');
   write (Bias, OptimalFirstGroupMember [g,d]:3, ' to cellobject ');
   writeln (Bias, OptimalLastGroupMember [g,d]:3);
END; {For g}
writeln (Bias);

END; {OutputOfOptimalDivision}

{-------------------------------------------------------------------------}
{-------------------------------------------------------------------------}

PROCEDURE CalculationOfGroupRatio (FirstMember: integer;
                                   LastMember:  integer);

TYPE MethodVector = ARRAY [1..NumberOfMatrixMethods] OF integer;

VAR GroupApportionment: MethodVector;

BEGIN {CalculationOfGroupRatio}

GroupFairShare:= 0;
FOR p:= 1 TO NumberOfMatrixMethods DO
   GroupApportionment [p] := 0;

FOR l:= FirstMember TO LastMember DO BEGIN
   GroupFairShare:= GroupFairShare + Cell [l].FairShare;
   FOR p:= 1 TO NumberOfMatrixMethods DO
      GroupApportionment [p] := GroupApportionment [p] + MatrixApportionments
         [Cell [l].ConstituencyNumber, Cell [l].PartyNumber, p];
END; {For l}

FOR p:= 1 TO NumberOfMatrixMethods DO
   IF (GroupFairShare = 0) AND (GroupApportionment [p] = 0) THEN
      GroupRatio [p] := 1
   ELSE GroupRatio [p] := GroupApportionment [p] / GroupFairShare;

END; {CalculationOfGroupRatio}

{-------------------------------------------------------------------------}
{-------------------------------------------------------------------------}
{-------------------------------------------------------------------------}
```

```
BEGIN {MatrixBias}

{Section for reading of data}

rewrite (Bias);

MethodName [1] := 'LF ';
MethodName [2] := 'SD ';
MethodName [3] := 'DM ';
MethodName [4] := 'MF ';
MethodName [5] := 'HA ';

ReadingOfElectionData (Ice63);

CalculationOfSums (Votes, ConstituencyVotes, PartyVotes, SumVotes);

Optimal:= True;
ReadingOfGamsData (TwoDimLFGamsOutput);

writeln (Bias, 'Inputfile Ice63');
writeln (Bias);
writeln (Bias, 'LF matrix apportionment error messages ');
writeln (Bias);
IF NOT(Optimal) THEN
   writeln (Bias, 'The two dimensional LF solution is not optimal ');

FOR i:= 1 TO NumberOfConstituencies DO
   FOR j:= 1 TO NumberOfParties DO
      WholeSeats [i,j] := (Votes [i,j] * NumberOfSeats) DIV SumVotes;

FOR i:= 1 TO NumberOfConstituencies DO
   FOR j:= 1 TO NumberOfParties DO BEGIN
      MatrixApportionments [i,j,1] := FractionalSeats [i,j] + WholeSeats [i,j];
      LFApportionment [i,j] := MatrixApportionments [i,j,1];
   END; {For j}

CalculationOfSums (WholeSeats, WholeConstituencySeats,
                   WholePartySeats, SumWholeSeats);

CalculationOfSums (FractionalSeats, FractionalConstituencySeats,
                   FractionalPartySeats, SumFractionalSeats);

FOR i:= 1 TO NumberOfConstituencies DO
   ConstituencyRepresentatives [i] :=
      WholeConstituencySeats [i] + FractionalConstituencySeats [i];

FOR j:= 1 TO NumberOfParties DO
   PartyRepresentatives [j] :=
      WholePartySeats [j] + FractionalPartySeats [j];

IF (SumWholeSeats + SumFractionalSeats) <> NumberOfSeats THEN
   writeln (Bias, 'The seat sum is wrong ');
WritingOfLine (Bias);
writeln (Bias);

FOR p:= 2 TO NumberOfMatrixMethods DO
   ReadingOfRepresentativesData (BiasApport);

writeln (Bias, 'Matrix apportionment with LF ');
OutputOfIntegerMatrices (Bias, 5, 6, LFApportionment);

reset (BiasApport);
WHILE NOT (EOF(BiasApport)) DO BEGIN
   WHILE NOT (EOLN(BiasApport)) DO BEGIN
      read (BiasApport, Character);
      write (Bias, Character);
   END; {While eoln}
   readln (BiasApport);
   writeln (Bias);
END; {While eof}

WritingOfLine (Bias);
writeln (Bias);

{------------------------------------------------------------------------------}
```

```
{Section for finding the fair share allocation}

FOR i:= 1 TO NumberOfConstituencies DO
   FOR j:= 1 TO NumberOfParties DO
      InitialFairShares [i,j] := (Votes [i,j] * NumberOfSeats) / SumVotes;

writeln (Bias, 'Initial fair shares ');
OutputOfRealMatrices (Bias, 7, 9, 2, InitialFairShares);

Discrepancy:= 100;
DoubleIterations:= 0;

FOR i:= 1 TO NumberOfConstituencies DO
   ContinuousConstituencyMultiplie [i] := 1;

FOR j:= 1 TO NumberOfParties DO
   ContinuousPartyMultiplier [j] := 1;

WHILE Discrepancy > AllowedDiscrepancy DO BEGIN
   Discrepancy:= 0;
   DoubleIterations:= DoubleIterations + 1;
   ScalingOfContinuousMultipliers;
END; {While}

FOR i:= 1 TO NumberOfConstituencies DO
   FOR j:= 1 TO NumberOfParties DO
      FinalFairShares [i,j] := InitialFairShares [i,j] *
        ContinuousConstituencyMultiplie [i] * ContinuousPartyMultiplier [j];

writeln (Bias, 'Number of double iterations to find the final fair shares ',
   DoubleIterations:3);
writeln (Bias, 'Discrepancy ', Discrepancy:9:6);
writeln (Bias);
writeln (Bias, 'Final fair shares ');
OutputOfRealMatrices (Bias, 7, 9, 2, FinalFairShares);
WritingOfLine (Bias);

{-------------------------------------------------------------------------------}

{Section for calculation of seat differences}

FOR p:= 1 TO NumberOfMatrixMethods DO
   FOR SecondMethod:= 1 TO NumberOfMatrixMethods DO
      SeatDifference [p,SecondMethod] := 0;

FOR p:= 1 TO NumberOfMatrixMethods DO
   FOR SecondMethod:= 1 TO p DO
      FOR i:= 1 TO NumberOfConstituencies DO
         FOR j:= 1 TO NumberOfParties DO
            SeatDifference [p,SecondMethod] := SeatDifference [p,SecondMethod]
               +  ABS(MatrixApportionments [i,j,p] -
                     MatrixApportionments [i,j,SecondMethod]);;

writeln (Bias);
writeln (Bias, 'Number of seats placed differently ');
writeln (Bias);
write (Bias, '     ');
FOR SecondMethod:= 1 TO NumberOfMatrixMethods DO
   write (Bias, MethodName [SecondMethod], ' ');
writeln (Bias);
FOR p:= 1 TO NumberOfMatrixMethods DO BEGIN
   write (Bias, MethodName [p]);
   FOR SecondMethod:= 1 TO p DO
      write (Bias, (SeatDifference [p,SecondMethod] DIV 2):4);
   writeln (Bias);
END; {For p}
writeln (Bias);
WritingOfLine (Bias);
WritingOfLine (Bias);

{-------------------------------------------------------------------------------}
```

```
{Section for cellobjects}

{Because of "strange" apportionments for the largest cells, we divide the
 parties in two classes based on a DividingFactor multiplied by the ratio
 (PartyRepresentatives / NumberOfConstituencies) and calculate the biases.}

NumberOfSelectedParties:= 0;
FOR j:= 1 TO NumberOfParties DO
   IF PartyRepresentatives [j] > (DividingFactor * NumberOfConstituencies) THEN
      NumberOfSelectedParties:= NumberOfSelectedParties + 1;
IF (NumberOfSelectedParties = 0) OR
   (NumberOfSelectedParties = NumberOfParties) THEN
   NumberOfPartyDivisions:= 1
ELSE NumberOfPartyDivisions:= 3;

f:= 1;
WHILE f <= NumberOfPartyDivisions DO BEGIN
   FOR j:= 1 TO NumberOfParties DO
      SelectedParty [j] := False;

   CASE f OF
      1: ;
      2, 3: BEGIN
            FOR l:= 1 TO (NumberOfConstituencies * NumberOfParties) DO BEGIN
               Cell [l].Name:= 'Z';
               Cell [l].FairShare:= 0;
               Cell [l].ConstituencyNumber:= 1;
               Cell [l].PartyNumber:= 1;
            END; {For l}
            NumberOfSelectedParties:= 0;
            NumberOfSelectedSeats:= 0;
            END;
   END; {Case f}

   CASE f OF
      1: FOR j:= 1 TO NumberOfParties DO
            SelectedParty [j] := True;
      2: FOR j:= 1 TO NumberOfParties DO
            IF PartyRepresentatives [j] >
              (DividingFactor * NumberOfConstituencies) THEN
               SelectedParty [j] := True;
      3: FOR j:= 1 TO NumberOfParties DO
            IF PartyRepresentatives [j] <=
              (DividingFactor * NumberOfConstituencies) THEN
               SelectedParty [j] := True;
   END; {Case f}

   CASE f OF
      1: BEGIN
            NumberOfSelectedParties:= NumberOfParties;
            NumberOfSelectedSeats:= NumberOfSeats;
         END;
      2, 3: FOR j:= 1 TO NumberOfParties DO
               IF SelectedParty [j] THEN BEGIN
                  NumberOfSelectedParties:= NumberOfSelectedParties + 1;
                  NumberOfSelectedSeats:=
                     NumberOfSelectedSeats + PartyRepresentatives [j];
               END; {Then}
   END; {Case f}

   rewrite (TemporaryFile);
   l:= 1;
   FOR i:= 1 TO NumberOfConstituencies DO
      FOR j:= 1 TO NumberOfParties DO
         IF SelectedParty [j] THEN BEGIN
            write (TemporaryFile, ConstituencyName [i]);
            write (TemporaryFile, ' ');
            writeln (TemporaryFile, PartyName [j]);
            reset (TemporaryFile);
            readln (TemporaryFile, Cell [l].Name);
            rewrite (TemporaryFile);
            Cell [l].FairShare:= FinalFairShares [i,j];
            Cell [l].ConstituencyNumber:= i;
            Cell [l].PartyNumber:= j;
            l:= l + 1;
         END; {Then}
```

```
   NumberOfCellObjects:= NumberOfConstituencies * NumberOfSelectedParties;
   SortingOfCells (1, NumberOfCellObjects);
   writeln (Bias);
   write (Bias, 'Sorting of cellobjects');
   CASE f OF
      1: writeln (Bias, ' from all parties');;
      2: writeln (Bias, ' from large parties');
      3: writeln (Bias, ' from small parties');
   END; {Case f}
   writeln (Bias);
   write (Bias, '                          ');
   FOR p:= 1 TO NumberOfMatrixMethods DO
      write (Bias, MethodName [p], ' ');
   writeln (Bias);
   FOR l:= 1 TO NumberOfCellObjects DO BEGIN
      write (Bias, l:3, ' ', Cell [l].Name, '   ', Cell [l].FairShare:8:5);
      FOR p:= 1 TO NumberOfMatrixMethods DO
         write (Bias, MatrixApportionments [Cell [l].ConstituencyNumber,
                                            Cell [l].PartyNumber, p] :4);
      writeln (Bias);
   END; {For l}
   writeln (Bias);
   WritingOfLine (Bias);

{-------------------------------------------------------------------------------}

{Section for divisions}

   DivisionName [1] := 'Number ';
   DivisionName [2] := 'Size   ';
   DivisionName [3] := 'Cluster';
   DivisionName [4] := 'Quota  ';

   FOR NumberOfGroups:= 2 TO MIN(NumberOfCellObjects, MaxNumberOfGroups)
      DO BEGIN

      writeln (Bias);
      writeln (Bias, 'Error messages for division in ', NumberOfGroups:1,
         ' groups ');
      writeln (Bias);

      FOR g:= 1 TO NumberOfGroups DO
         FOR d:= 1 TO 4 DO BEGIN
            OptimalFirstGroupMember [g,d] := 0;
            OptimalLastGroupMember [g,d] := 0;
         END; {For d}

      FOR d:= 1 TO 4 DO BEGIN
         OptimalFirstGroupMember [1,d] := 1;
         OptimalLastGroupMember [NumberOfGroups,d] := NumberOfCellObjects;
      END; {For d}

      d:= 1;
      NumberDivision;

      d:= 2;
      EmptySizeDivision:= False;
      SizeDivision;

      d:= 3;
      ClusterDivision;

      d:= 4;
      QuotaDivision (NumberOfSelectedSeats);

{-------------------------------------------------------------------------------}
```

```
{Section for bias calculation}

      writeln (Bias);
      writeln (Bias, 'Bias percentages with ', NumberOfGroups:1, ' groups ');
      writeln (Bias);
      WritingOfLine (Bias);

      FOR d:= 1 TO 4 DO BEGIN
         writeln (Bias);
         OutputOfOptimalDivision;

         IF NOT ((d = 2) AND EmptySizeDivision) THEN BEGIN
            FOR g:= 1 TO NumberOfGroups DO BEGIN
               CalculationOfGroupRatio (OptimalFirstGroupMember [g,d],
                                        OptimalLastGroupMember [g,d]);
               FOR p:= 1 TO NumberOfMatrixMethods DO
                  GroupRatios [g,d,p] := GroupRatio [p];
            END; {For g}

            FOR p:= 1 TO NumberOfMatrixMethods DO
               FOR g:= 1 TO (NumberOfGroups - 1) DO
                  FOR SecondGroup:= (g + 1) TO NumberOfGroups DO
                     IF GroupRatios [g,d,p] = 0 THEN
                        IF GroupRatios [SecondGroup,d,p] = 0 THEN
                           BiasPercentage [g,SecondGroup,d,p] := -888.88
                        ELSE BiasPercentage [g,SecondGroup,d,p] := -999.99
                     ELSE BiasPercentage [g,SecondGroup,d,p] :=
                         (100 * (GroupRatios [g,d,p] -
                          GroupRatios [SecondGroup,d,p])) /
                          GroupRatios [g,d,p];

            write (Bias, '              ');
            FOR p:= 1 TO NumberOfMatrixMethods DO
               write (Bias, MethodName [p], '       ');
            writeln (Bias);

            FOR g:= 1 TO (NumberOfGroups - 1) DO
               FOR SecondGroup:= (g + 1) TO NumberOfGroups DO BEGIN
                  write (Bias, 'E(', g:1, ',', SecondGroup:1, ')  ');
                  FOR p:= 1 TO NumberOfMatrixMethods DO BEGIN
                     write (Bias, BiasPercentage [g,SecondGroup,d,p] :7:2);
                     write (Bias, '%  ');
                  END; {For p}
                  writeln (Bias);
               END; {For SecondGroup}
            writeln (Bias);
         END; {Then}
         WritingOfLine (Bias);
      END; {For d}
   END; {For NumberOfGroups}

   WritingOfLine (Bias);
   f:= f + 1;
END; {While f}

END. {MatrixBias}
```

Appendix 3:

Data regarding the apportionment algorithm

Description of Data sets tables

These tables show the following basic information about the data sets utilized in the algorithm tests:

- Country
- Year of election
- Number of constituencies (m)
- Number of parties (n)
- House size (h)

The following more specialized information is also included:

- Number of parties which only participate in one constituency
- How many constituencies (> 1) the "next" party participates in
- How many constituencies the "next" party participates in as a percentage of the total number of constituencies

The last column in the tables shows the charged CPU time for solving the 2 x 2 x 3 x 3 x 4 = 144 cases for each data set.

NB! these CPU times have been achieved with a somewhat simpler version of the program ElectionAlgorithm than the one presented in Appendix 2.

Data sets

Austria				Information about regional parties			
				# of parties which only participate in one constituency	# of constituencies the next party participates in and its participation ratio		Charged CPU Halfrun.Com
Year	*m*	*n*	*h*				
1994	9	5	183	0	9	100 %	0:17.36
1990	9	5	183	0	9	100 %	0:17.82
1986	9	4	183	0	9	100 %	0:15.96
1983	9	5	183	0	9	100 %	0:17.83

Denmark				Information about regional parties			
				# of parties which only participate in one constituency	# of constituencies the next party participates in and its participation ratio		Charged CPU Halfrun.Com
Year	*m*	*n*	*h*				
1994	17	9	174	0	17	100 %	1:16.75
1990	17	10	175	0	17	100 %	1:24.23
1988	17	10	175	0	17	100 %	1:31.16
1987	17	11	175	0	17	100 %	1:42.00
1984	17	10	175	0	17	100 %	1:15.93
1981	17	10	175	0	17	100 %	1:19.43
1979	17	11	175	0	17	100 %	1:39.29
1977	17	11	175	0	17	100 %	1:43.43
1975	17	11	175	0	17	100 %	1:38.33
1973	17	11	175	0	17	100 %	1:39.73
1971	17	9	175	0	17	100 %	1:01.84
1968	23	8	175	0	23	100 %	1:12.16
1966	23	7	175	0	23	100 %	1:24.74

Data sets

Finland				Information about regional parties			
				# of parties which only participate in one constituency	# of constituencies the next party participates in and its participation ratio		Charged CPU Halfrun.Com
Year	*m*	*n*	*h*				
1995	14	10	199	0	4	29 %	1:05.50
1991	14	9	199	0	4	29 %	0:50.28
1987	14	10	199	0	7	50 %	1:19.17
1983	14	8	196	0	3	21 %	0:43.47
1979	14	9	199	0	5	36 %	0:47.92
1975	14	10	199	0	5	36 %	1:09.78
1972	15	7	196	0	5	33 %	0:39.47
1970	15	7	199	0	5	33 %	0:42.79
1966	15	7	199	0	5	33 %	0:46.27
1962	15	7	199	0	5	33 %	0:41.95
1958	16	7	200	0	4	25 %	0:40.63
1954	16	6	200	0	6	38 %	0:41.76
1951	15	6	200	0	5	33 %	0:33.85
1948	15	6	200	0	5	33 %	0:35.55
1945	15	8	200	0	2	13 %	0:43.92
1939	15	7	200	0	4	27 %	0:47.81
1936	16	8	200	0	2	13 %	0:53.55

(West) Germany				Information about regional parties			
				# of parties which only participate in one constituency	# of constituencies the next party participates in and its participation ratio		Charged CPU Halfrun.Com
Year	*m*	*n*	*h*				
1994	16	5	672	0	16	100 %	1:40.84
1990	16	6	662	0	6	38 %	2:13.21
1987	10	4	497	0	10	100 %	0:29.96
1983	10	4	498	0	10	100 %	0:34.72
1980	10	4	497	0	10	100 %	0:28.15
1976	10	3	496	0	10	100 %	0:24.10
1972	10	3	496	0	10	100 %	0:22.70
1969	10	4	496	0	10	100 %	0:33.12
1965	10	4	496	0	10	100 %	0:32.47
1961	10	4	499	0	10	100 %	0:33.76
1957	10	5	497	0	10	100 %	0:45.14
1953	9	10	487	2	6	67 %	1:30.11
1949	9	10	399	3	3	33 %	0:55.11
1919	36	13	421	5	2	6 %	9:15.75

Data sets

Iceland				Information about regional parties			
				# of parties which only participate in one constituency	# of constituencies the next party participates in and its participation ratio		Charged CPU Halfrun.Com
Year	*m*	*n*	*h*				
1995	8	6	63	0	8	100 %	0:11.20
1991	8	7	63	0	8	100 %	0:11.96
1987	8	9	63	1	5	63 %	0:14.52
1983	8	6	60	0	3	38 %	0:11.10
1979	8	5	60	1	8	100 %	0:09.70
1978	8	5	60	0	8	100 %	0:09.75
1974	8	5	60	0	8	100 %	0:09.99
1971	8	6	60	0	3	38 %	0:11.37
1967	8	6	60	1	2	25 %	0:10.30
1963	8	4	60	0	8	100 %	0:09.18
1959	8	5	60	0	3	38 %	0:09.63

Luxembourg				Information about regional parties			
				# of parties which only participate in one constituency	# of constituencies the next party participates in and its participation ratio		Charged CPU Halfrun.Com
Year	*m*	*n*	*h*				
1959	4	4	52	0	2	50 %	0:07.81
1954	4	5	52	0	2	50 %	0:08.06
1948/5	4	4	52	0	4	100 %	0:07.73
1945	4	6	51	2	4	100 %	0:08.12
1934/3	4	10	55	7	2	50 %	0:08.93

Norway				Information about regional parties			
				# of parties which only participate in one constituency	# of constituencies the next party participates in and its participation ratio		Charged CPU Halfrun.Com
Year	*m*	*n*	*h*				
1993	19	8	165	0	19	100 %	1:13.46
1989	19	8	165	1	19	100 %	1:05.63
1985	19	7	157	0	19	100 %	0:52.72
1981	19	10	155	2	4	21 %	1:10.85
1977	19	12	155	3	2	11 %	1:07.13
1973	19	11	155	2	8	42 %	1:27.08
1969	20	8	150	1	4	20 %	1:00.20
1965	20	9	150	2	15	75 %	1:15.76
1961	20	9	150	1	5	25 %	1:08.06
1957	20	8	150	1	2	10 %	0:55.17
1953	20	7	150	1	15	75 %	0:53.05
1949	29	9	150	0	2	7 %	1:36.93

Data sets

Sweden				Information about regional parties			
				# of parties which only participate in one constituency	# of constituencies the next party participates in and its participation ratio		Charged CPU Halfrun.Com
Year	*m*	*n*	*h*				
1994	24	8	349	0	24	100 %	3:04.15
1991	28	8	349	0	28	100 %	3:36.77
1988	28	7	349	0	28	100 %	3:20.37
1985	28	6	349	0	28	100 %	1:59.72
1982	28	6	349	0	28	100 %	2:22.35
1979	28	6	349	0	28	100 %	2:21.77
1976	28	6	349	0	28	100 %	2:45.92
1973	28	6	350	0	28	100 %	2:28.55
1970	28	6	350	0	28	100 %	2:15.33
1968	28	8	233	1	2	7 %	1:57.00
1964	28	7	233	1	25	89 %	1:53.04
1960	28	5	232	0	28	100 %	1:07.84
1958	28	5	231	0	18	64 %	1:12.41
1956	28	5	231	0	28	100 %	1:21.63
1952	28	5	230	0	28	100 %	1:21.07
1948	28	5	230	0	27	96 %	1:12.50
1944	28	5	230	0	26	93 %	1:19.92
1940	28	5	230	0	26	93 %	1:23.32
1936	28	6	230	0	27	96 %	1:36.21
1932	28	6	230	0	11	39 %	1:49.86
1928	28	6	230	0	16	57 %	1:30.14
1924	28	7	230	0	22	79 %	2:02.96
1921	28	6	230	0	17	61 %	1:33.16
1920	56	7	228	0	2	4 %	7:01.82
1917	56	5	230	0	38	68 %	3:48.00
1914 II	56	3	230	0	56	100 %	1:26.77
1914 I	56	3	230	0	55	98 %	1:25.38
1911	56	3	230	0	55	98 %	1:35.05

Description of Algorithm data tables

We have operated with 2 Relaxations, 2 Selection methods, 3 Initialization procedures, 3 Bound vectors, and 4 Matrix apportionment methods for each data set, i.e. the algorithm has been run a total of 144 times. Algorithm data for each country have been splitted in two based on Relaxation. Each Relaxation table consists of 4 main parts: Selection, Initialization, Bound vector, and Matrix method. In addition there is a column called Goodness. Each of the main parts consists of several columns. Here follows a description of the contents of the different columns. We look at the column headings in the third row, i.e. to the right of "Year":

Goodness columns:

Show the average initial measure of goodness for the specified combination. The Selection used does not influence the initial measure of goodness. The averages in the column called Goodness and with "Initial" in the third row are therefore averages over 3 x 3 x 4 = 36 combinations. The averages in the main parts are calculated from the following number of cases: 3 x 4 = 12 in the Initialization and Bound vector parts and 3 x 3 = 9 in the Matrix method part.

Iteration columns:

Show the average number of iterations used to solve the matrix apportionment problem for the specified combination. These averages are calculated from the following number of cases: 3 x 3 x 4 = 36 in the Selection part, 2 x 3 x 4 = 24 in the Initialization and Bound vector parts, and 2 x 3 x 3 = 18 in the Matrix method part.

Ratio columns (in the Selection part):

Show the average decrease in the measure of goodness per iteration. The averages are calculated from 3 x 3 x 4 = 36 cases.

Sol. columns (in the Initialization part):

Show the number of direct solutions for the data set with the specified Initialization and Relaxation. There are 3 x 4 = 12 combinations of Bound vector and Matrix method, so 12 is the highest achievable number of direct solutions.

Sum row:

Shows the total number of direct solutions with the specified Initialization.

Average row:

Shows the average of the average data set figures.

For further comments regarding Algorithm data see chapter 14.

AUSTRIA (Algorithm data)

Constituency relaxation

	Goodness	Selection				Initialization								Bound vector						Matrix method							
	Whole data set	Representation		ρ-effect		No			Quota ratio			Apportionment			EL-EL		MF-MF		SD-HA		$CP_{0,01}$		DM		MF		HA	
Year	Initial	Iterations	Ratio	Iterations	Ratio	Goodness	Iterations	Sol.	Goodness	Iterations	Sol.	Goodness	Iterations	Sol.	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations
1994	8,50	7,03	1,23	5,94	1,43	12,00	7,08	0	9,67	7,67	0	3,83	4,71	2	9,67	7,46	7,67	6,13	8,17	5,88	11,33	6,06	5,78	6,72	4,67	4,44	12,22	8,72
1990	7,67	7,08	1,10	5,75	1,42	10,83	8,13	0	8,00	7,13	0	4,17	4,00	0	8,67	6,29	7,33	6,71	7,00	6,25	9,56	7,83	5,11	5,28	5,56	4,83	10,44	7,72
1986	6,67	7,53	0,90	6,69	1,07	8,67	8,17	0	7,50	8,13	0	3,83	5,04	0	6,00	5,75	7,17	8,71	6,83	6,88	7,11	7,78	6,00	6,39	5,11	6,39	8,44	7,89
1983	5,22	5,50	1,08	4,17	1,41	5,67	5,50	0	4,50	4,29	0	5,50	4,71	0	5,17	4,54	5,67	5,67	4,83	4,29	7,56	5,56	2,67	2,44	3,56	4,89	7,11	6,44
Sum								0			0			2														
Average	7,02	6,79	1,08	5,64	1,33	9,29	7,22		7,42	6,81		4,33	4,62		7,38	6,01	6,96	6,81	6,71	5,83	8,89	6,81	4,89	5,21	4,73	5,14	9,55	7,69

Party relaxation

	Goodness	Selection				Initialization								Bound vector						Matrix method							
	Whole data set	Representation		ρ-effect		No			Quota ratio			Apportionment			EL-EL		MF-MF		SD-HA		$CP_{0,01}$		DM		MF		HA	
Year	Initial	Iterations	Ratio	Iterations	Ratio	Goodness	Iterations	Sol.	Goodness	Iterations	Sol.	Goodness	Iterations	Sol.	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations
1994	6,94	4,22	1,81	2,94	2,50	9,33	4,29	0	9,33	4,25	1	2,17	2,21	1	7,67	4,63	7,17	3,67	6,00	2,46	10,67	3,50	4,67	3,78	2,44	3,00	10,00	4,06
1990	9,67	4,56	2,18	3,17	3,02	13,50	4,54	0	12,17	4,58	0	3,33	2,46	0	9,83	3,08	9,50	4,42	9,67	4,08	15,78	3,89	5,33	3,50	5,11	3,17	12,44	4,89
1986	6,94	3,58	2,33	2,69	2,98	8,67	3,92	0	8,50	3,50	0	3,67	2,00	0	6,67	2,63	7,00	3,71	7,17	3,08	11,78	2,94	4,89	3,83	3,78	2,44	7,33	3,33
1983	11,50	8,00	1,98	2,47	4,16	16,50	7,08	0	14,33	6,67	0	3,67	1,96	0	12,67	4,21	10,83	5,96	11,00	5,54	24,67	10,94	5,56	2,44	4,44	3,06	11,33	4,50
Sum								0			1			1														
Average	8,76	5,09	2,08	2,82	3,17	12,00	4,96		11,08	4,75		3,21	2,16		9,21	3,64	8,63	4,44	8,46	3,79	15,73	5,32	5,11	3,39	3,94	2,92	10,28	4,20

DENMARK (Algorithm data)

Constituency relaxation

	Goodness	Selection				Initialization									Bound vector						Matrix method							
	Whole data set	Representation		ρ-effect		No			Quota ratio			Apportionment			EL-EL		MF-MF		SD-HA		$CP_{0.01}$		DM		MF		HA	
Year	Initial	Iterations	Ratio	Iterations	Ratio	Goodness	Iterations	Sol.	Goodness	Iterations	Sol.	Goodness	Iterations	Sol.	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations
1994	18,28	20,17	0,87	16,69	1,07	17,17	17,46	0	16,00	18,04	0	21,67	19,79	0	18,00	18,83	18,33	18,38	18,50	18,08	26,67	23,17	11,11	15,00	12,44	15,61	22,89	19,94
1990	20,17	19,31	1,05	15,92	1,25	20,00	17,46	0	18,83	17,50	0	21,67	17,88	0	18,83	15,96	20,67	18,25	21,00	18,63	26,44	20,17	12,44	13,67	15,11	18,56	26,67	18,06
1988	19,83	19,58	1,02	18,06	1,18	21,17	19,33	0	20,00	18,42	0	18,33	18,71	0	17,67	16,58	20,33	22,04	21,50	17,83	26,89	19,22	11,56	17,94	14,44	19,78	26,44	18,33
1987	21,11	20,83	1,00	18,89	1,12	21,17	19,71	0	19,67	18,46	0	22,50	21,42	0	18,33	16,63	21,00	22,29	24,00	20,67	29,78	21,22	12,44	17,83	13,56	20,56	28,67	19,83
1984	18,78	16,25	1,13	14,81	1,23	18,17	14,92	0	15,50	15,67	0	22,67	16,00	0	17,83	11,04	19,00	19,92	19,50	15,63	30,89	19,17	8,44	11,94	11,33	11,28	24,44	19,72
1981	17,17	17,89	0,94	14,92	1,12	19,50	17,04	0	16,00	15,88	0	16,00	16,29	0	16,83	16,71	16,83	15,46	17,83	17,04	28,67	19,89	7,56	10,50	10,22	11,83	22,22	23,39
1979	22,33	20,28	1,17	20,47	1,22	22,67	20,08	0	20,00	20,92	0	24,33	20,13	0	20,50	14,42	23,67	26,17	22,83	20,54	34,00	34,28	13,78	13,89	14,44	11,78	27,11	21,56
1977	22,11	20,17	1,06	19,75	1,08	20,17	19,33	0	20,17	18,92	0	26,00	21,63	0	21,67	17,33	22,83	25,00	21,83	17,54	35,11	23,61	11,78	18,67	14,44	18,33	27,11	19,22
1975	21,94	20,06	1,07	18,94	1,13	21,50	19,21	0	18,67	18,50	0	25,67	20,79	0	19,00	16,50	24,33	21,21	22,50	20,79	30,89	20,72	12,67	18,33	14,67	16,61	29,56	22,33
1973	18,50	20,14	0,95	18,61	1,05	20,67	19,33	0	18,00	16,92	0	16,83	21,88	0	15,83	19,25	20,50	19,00	19,17	19,88	25,11	24,67	10,22	15,78	13,33	15,00	25,33	22,06
1971	16,28	15,42	1,12	14,03	1,25	17,33	16,17	0	14,00	13,88	0	17,50	14,13	0	14,17	10,17	17,50	18,71	17,17	15,29	25,11	20,50	10,67	9,56	11,33	10,56	18,00	18,28
1968	15,50	15,53	1,04	13,97	1,15	19,00	16,17	0	12,67	12,71	0	14,83	15,38	0	15,33	11,08	16,00	15,92	15,17	17,25	23,78	24,06	9,11	11,44	8,67	10,44	20,44	13,06
1966	19,83	24,97	0,85	23,14	0,89	22,00	23,75	0	18,50	22,00	0	19,00	26,42	0	19,83	21,08	19,50	27,04	20,17	24,04	26,00	26,67	12,44	18,06	14,89	19,28	26,00	32,22
Sum								0			0			0														
Average	19,37	19,28	1,02	17,55	1,13	20,04	18,46		17,54	17,52		20,54	19,27		17,99	15,81	20,04	20,72	20,09	18,71	28,41	22,87	11,09	14,82	12,99	15,36	24,99	20,62

Party relaxation

	Goodness	Selection				Initialization									Bound vector						Matrix method							
	Whole data set	Representation		ρ-effect		No			Quota ratio			Apportionment			EL-EL		MF-MF		SD-HA		$CP_{0.01}$		DM		MF		HA	
Year	Initial	Iterations	Ratio	Iterations	Ratio	Goodness	Iterations	Sol.	Goodness	Iterations	Sol.	Goodness	Iterations	Sol.	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations
1994	27,61	12,58	1,81	10,33	2,30	38,33	13,04	0	38,83	14,13	0	5,67	7,21	0	26,33	10,46	28,00	12,04	28,50	11,88	53,78	15,11	11,56	8,72	10,89	9,17	34,22	12,83
1990	29,83	14,83	1,79	11,61	2,36	41,50	16,17	0	40,50	15,13	0	7,50	8,38	0	27,67	11,33	30,83	14,46	31,00	13,88	57,56	16,22	12,00	10,11	14,44	12,11	35,33	14,44
1988	26,33	16,11	1,53	13,92	1,78	36,00	17,25	0	36,00	16,96	0	7,00	10,83	0	23,67	10,67	27,33	19,25	28,00	15,13	49,78	15,44	12,00	14,00	12,67	13,78	30,89	16,83
1987	31,39	16,92	1,66	14,58	1,88	44,17	18,75	0	43,17	19,67	0	6,83	8,83	0	28,67	12,96	32,17	16,71	33,33	17,58	60,00	17,61	14,67	13,39	16,22	15,44	34,67	16,56
1984	29,39	12,28	2,16	11,11	2,54	41,00	14,33	0	41,50	13,92	0	5,67	6,83	0	28,00	7,71	29,67	15,25	30,50	12,13	56,44	14,94	10,00	9,89	10,89	7,72	40,22	14,22
1981	24,33	13,36	1,68	11,22	2,08	33,50	14,46	0	32,50	14,42	0	7,00	8,00	0	24,17	11,46	23,67	12,00	25,17	13,42	44,44	13,78	13,11	9,17	12,00	9,06	27,78	17,17
1979	31,22	15,08	1,95	13,06	2,28	42,17	15,83	0	41,67	15,71	0	9,83	10,67	0	27,33	10,42	33,17	18,38	33,17	13,42	56,89	23,11	13,78	9,50	15,78	8,44	38,44	15,22
1977	32,22	17,17	1,84	15,00	2,07	44,83	17,67	0	44,50	17,79	0	7,33	12,79	0	30,33	13,38	33,50	20,00	32,83	14,88	55,11	16,06	14,44	16,56	13,33	11,44	46,00	20,28
1975	33,00	15,47	1,96	14,31	2,09	45,17	18,17	0	45,17	16,42	0	8,67	10,08	0	31,00	12,79	34,50	16,92	33,50	14,96	59,33	16,72	17,11	14,06	14,44	9,94	41,11	18,83
1973	24,06	16,00	1,43	14,11	1,61	32,00	18,63	0	31,50	16,04	0	8,67	10,50	0	21,83	13,50	25,33	16,25	25,00	15,42	38,22	18,22	13,78	12,61	13,33	11,72	30,89	17,67
1971	26,33	10,31	2,45	7,50	3,04	38,00	11,04	0	35,17	11,29	0	5,83	4,38	0	24,83	5,54	27,17	11,42	27,00	9,75	64,44	13,00	10,67	5,33	10,44	6,00	19,78	11,28
1968	21,94	11,25	1,78	7,53	2,69	30,83	11,42	0	28,83	11,04	0	6,17	5,71	0	20,00	6,38	22,67	9,25	23,17	12,54	53,78	13,11	6,00	6,06	9,11	8,00	18,89	10,39
1966	26,89	10,92	2,46	8,47	2,98	35,67	10,46	0	35,33	11,00	0	9,67	7,63	0	25,83	8,63	27,00	11,29	27,83	9,17	51,78	10,94	14,22	7,83	14,00	7,94	27,56	12,06
Sum								0			0			0														
Average	28,04	14,02	1,88	11,75	2,28	38,71	15,17		38,05	14,89		7,37	8,60		26,13	10,40	28,85	14,86	29,15	13,40	53,97	15,71	12,56	10,56	12,89	10,06	32,75	15,21

FINLAND (Algorithm data)

Constituency relaxation

	Goodness	Selection				Initialization									Bound vector						Matrix method							
	Whole data set	Representation		ρ-effect		No			Quota ratio			Apportionment			EL-EL		MF-MF		SD-HA		$CP_{0,01}$		DM		MF		HA	
Year	Initial	Iterations	Ratio	Iterations	Ratio	Goodness	Iterations	Sol.	Goodness	Iterations	Sol.	Goodness	Iterations	Sol.	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations
1995	12,61	15,25	0,94	13,08	1,07	13,83	14,04	0	11,33	14,92	0	12,67	13,54	0	13,50	14,67	11,67	15,58	12,67	12,25	19,56	17,94	8,00	16,78	7,56	8,00	15,33	13,94
1991	12,17	11,92	1,03	11,22	1,14	13,50	11,83	0	12,00	10,96	0	11,00	11,92	1	13,67	13,75	11,67	10,83	11,17	10,13	18,00	13,39	10,00	10,72	7,11	8,28	13,56	13,89
1987	13,94	17,00	0,96	15,69	1,07	14,50	17,29	0	13,50	18,08	0	13,83	13,67	0	13,67	13,25	14,50	17,67	13,67	18,13	19,78	11,89	10,44	14,17	10,44	11,67	15,11	27,67
1983	8,44	11,00	0,89	9,78	1,03	10,17	12,38	0	8,33	10,46	0	6,83	8,33	0	9,33	12,67	8,17	7,17	7,83	11,33	12,22	13,56	6,44	7,56	5,56	7,83	9,56	12,61
1979	9,56	10,31	0,94	9,42	1,08	9,00	9,04	0	8,17	8,92	0	11,50	11,63	0	9,67	11,58	9,50	9,04	9,50	8,96	15,11	12,06	6,44	7,89	7,11	8,89	9,56	10,61
1975	13,78	14,69	1,01	13,08	1,12	14,00	14,38	0	12,33	13,58	0	15,00	13,71	0	13,33	10,13	14,00	14,00	14,00	17,54	19,33	15,39	12,22	11,61	10,89	12,00	12,67	16,56
1972	7,22	10,75	0,79	9,97	0,86	9,17	12,17	0	7,33	10,50	0	5,17	8,42	0	6,33	9,13	7,83	12,17	7,50	9,79	6,89	8,78	6,44	8,67	5,33	6,78	10,22	17,22
1970	8,67	12,61	0,71	10,97	0,88	11,50	12,96	0	9,17	11,29	0	5,33	11,13	0	9,50	13,88	8,00	9,63	8,50	11,88	10,22	12,83	7,33	11,61	5,33	9,33	11,78	13,39
1966	10,39	14,03	0,82	12,86	0,91	14,17	15,79	0	9,17	12,75	0	7,83	11,79	0	11,00	11,08	10,33	15,63	9,83	13,63	10,89	12,39	9,33	17,00	9,11	10,33	12,22	14,06
1962	11,44	12,33	0,98	11,72	1,04	14,67	13,79	0	11,83	12,96	0	7,83	9,33	0	12,33	12,63	11,67	12,25	10,33	11,21	12,67	11,89	10,44	11,50	10,00	9,83	12,67	14,89
1958	9,44	11,25	0,87	9,92	1,00	12,83	13,25	0	9,17	10,88	0	6,33	7,63	0	11,50	12,79	8,17	8,58	8,67	10,38	11,78	13,83	7,33	8,22	6,22	6,78	12,44	13,50
1954	10,00	13,67	0,85	13,53	0,85	13,00	16,08	0	10,83	14,67	0	6,17	10,04	0	10,17	10,17	10,00	16,96	9,83	13,67	12,67	13,28	8,44	17,94	6,67	8,67	12,22	14,50
1951	11,11	11,28	1,01	9,86	1,21	13,67	12,13	0	12,50	11,50	0	7,17	8,08	0	12,50	11,83	10,33	10,04	10,50	9,83	15,56	10,78	8,67	10,83	7,56	9,89	12,67	10,78
1948	10,06	10,94	1,01	10,08	1,14	14,00	12,38	0	10,50	10,67	0	5,67	8,50	0	10,50	11,42	10,00	9,50	9,67	10,63	13,33	11,50	6,22	8,61	8,44	8,78	12,22	13,17
1945	11,11	10,50	1,03	9,97	1,15	15,67	12,46	0	9,67	10,21	0	8,00	8,04	1	14,00	13,54	9,33	7,13	10,00	10,04	13,78	13,00	7,33	7,17	7,56	9,22	15,78	11,56
1939	11,06	12,56	0,95	11,14	1,12	14,33	14,17	0	9,67	9,67	0	9,17	11,71	0	12,00	13,17	10,83	9,08	10,33	13,29	14,44	11,83	8,44	12,28	8,44	13,00	12,89	10,28
1936	12,89	12,39	1,07	11,06	1,21	18,33	14,50	0	13,17	11,79	0	7,17	8,88	0	13,83	9,13	12,17	13,25	12,67	12,79	16,00	14,06	11,33	11,72	8,89	8,83	15,33	12,28
Sum								0			0			2														
Average	10,82	12,50	0,93	11,37	1,05	13,31	13,45		10,51	11,99		8,63	10,37		11,58	12,05	10,48	11,68	10,39	12,09	14,25	12,85	8,52	11,43	7,78	9,30	12,72	14,17

Party relaxation

	Goodness	Selection				Initialization									Bound vector						Matrix method							
	Whole data set	Representation		ρ-effect		No			Quota ratio			Apportionment			EL-EL		MF-MF		SD-HA		$CP_{0,01}$		DM		MF		HA	
Year	Initial	Iterations	Ratio	Iterations	Ratio	Goodness	Iterations	Sol.	Goodness	Iterations	Sol.	Goodness	Iterations	Sol.	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations
1995	23,06	11,58	1,73	9,50	2,40	32,67	12,29	0	31,17	13,25	0	5,33	6,08	0	24,67	8,29	21,83	11,92	22,67	11,42	51,33	14,22	10,89	8,89	8,44	6,33	21,56	12,72
1991	20,72	9,19	2,12	6,28	2,90	29,17	9,08	0	27,50	10,33	0	5,50	3,79	0	23,50	8,38	19,50	7,38	19,17	7,46	40,89	10,89	14,00	5,83	6,89	4,67	21,11	9,56
1987	26,39	16,42	1,45	12,86	2,29	38,33	17,08	0	35,17	16,25	0	5,67	10,58	0	30,17	10,54	24,17	17,92	24,83	15,46	54,44	14,89	15,78	9,94	9,78	9,94	25,56	23,78
1983	13,56	8,25	1,75	7,69	1,98	19,17	9,50	0	16,67	8,71	0	4,83	5,71	0	14,83	8,79	12,83	4,71	13,00	10,42	25,11	10,00	9,78	6,67	6,22	5,78	13,11	9,44
1979	20,89	9,97	1,96	6,83	2,93	30,33	10,08	0	27,83	9,63	0	4,50	5,50	0	21,67	10,33	20,33	8,17	20,67	6,71	42,44	9,22	13,11	6,89	7,33	7,72	20,67	9,78
1975	27,44	14,64	1,67	11,17	2,48	39,17	14,83	0	37,67	14,67	0	5,50	9,21	0	28,83	10,33	26,00	11,50	27,50	16,88	56,89	15,83	15,33	9,06	10,44	9,61	27,11	17,11
1972	8,06	6,78	1,49	5,89	1,69	10,83	6,96	0	9,50	6,88	0	3,83	5,17	0	9,33	6,21	7,50	6,33	7,33	6,46	11,11	4,78	6,22	5,67	4,67	3,72	10,22	11,17
1970	9,11	7,03	1,45	6,44	1,56	12,00	7,42	0	10,50	6,58	0	4,83	6,21	0	11,17	8,54	7,50	4,17	8,67	7,50	11,11	7,17	7,56	7,11	6,00	6,28	11,78	6,39
1966	13,83	7,61	1,78	7,14	2,00	18,67	9,04	0	17,33	7,25	0	5,50	5,83	0	15,17	6,21	13,83	7,75	12,50	8,17	21,33	7,22	11,56	7,72	6,89	4,89	15,56	9,67
1962	13,67	6,94	2,37	5,19	3,02	18,17	6,46	0	16,33	6,54	0	6,50	5,21	0	15,33	6,13	13,50	5,88	12,17	6,21	19,78	4,94	11,33	6,17	8,22	4,83	15,33	8,33
1958	9,61	6,61	1,62	5,28	2,05	12,00	6,58	0	10,83	6,29	0	6,00	4,96	0	10,67	5,67	8,67	5,50	9,50	6,67	15,78	7,44	7,33	6,11	4,89	4,50	10,44	5,72
1954	9,33	6,69	1,49	5,81	1,88	12,33	7,38	0	11,00	6,29	0	4,67	5,08	0	10,67	5,25	9,00	7,13	8,33	6,38	14,00	5,72	6,89	7,22	6,44	5,28	10,00	6,78
1951	9,22	5,67	1,69	4,72	2,17	11,67	6,04	0	11,00	5,42	0	5,00	4,13	0	10,83	5,04	8,17	5,29	8,67	5,25	14,00	4,39	7,56	6,78	4,67	3,94	10,67	5,67
1948	8,78	6,47	1,50	5,58	1,89	12,00	6,79	0	9,67	6,63	0	4,67	4,67	1	10,83	7,88	7,17	5,63	8,33	4,58	13,78	7,28	5,56	7,17	8,00	5,11	7,78	4,56
1945	13,17	7,06	1,86	6,03	2,23	16,33	6,67	1	16,50	6,96	0	6,67	6,00	0	14,33	8,13	12,17	4,71	13,00	6,79	26,44	7,39	5,78	4,28	8,44	6,44	12,00	8,06
1939	17,72	8,72	1,84	7,25	2,41	24,17	8,79	0	22,83	9,63	0	6,17	5,54	0	22,83	9,46	15,17	6,21	15,17	8,29	35,78	8,33	10,67	7,28	8,00	8,11	16,44	8,22
1936	18,67	8,42	2,00	6,67	2,55	25,00	8,67	0	24,17	9,00	0	6,83	4,96	0	24,17	7,42	15,33	7,42	16,50	7,79	34,00	10,94	12,89	7,11	8,44	5,06	19,33	7,06
Sum								1			0			1														
Average	15,48	8,71	1,75	7,08	2,26	21,29	9,04		19,75	8,84		5,41	5,80		17,59	7,80	14,27	7,51	14,59	8,38	28,72	8,86	10,13	7,05	7,28	6,01	15,80	9,65

(WEST) GERMANY (Algorithm data)

Constituency relaxation

	Goodness	Selection				Initialization									Bound vector						Matrix method							
	Whole data set	Representation		ρ-effect		No			Quota ratio			Apportionment			EL-EL		MF-MF		SD-HA		$CP_{0.01}$		DM		MF		HA	
Year	Initial	Iterations	Ratio	Iterations	Ratio	Goodness	Iterations	Sol.	Goodness	Iterations	Sol.	Goodness	Iterations	Sol.	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations
1994	13,50	12,67	1,07	11,11	1,26	19,50	14,38	0	14,50	12,42	0	6,50	8,88	0	16,50	12,92	12,17	10,38	11,83	12,38	15,33	11,44	10,44	13,22	7,78	7,67	20,44	15,22
1990	13,50	14,78	0,93	12,17	1,13	16,33	14,33	0	13,67	14,17	0	10,50	11,92	0	14,67	15,08	12,67	14,04	13,17	11,29	19,56	16,06	8,44	11,06	9,33	12,11	16,67	14,67
1987	6,67	5,94	1,10	5,00	1,29	8,50	6,79	0	8,83	6,63	0	2,67	3,00	3	6,00	4,29	6,83	7,17	7,17	4,96	9,11	7,83	3,11	4,06	2,44	1,94	12,00	8,06
1983	7,72	7,69	0,99	6,44	1,29	11,00	9,17	0	9,17	8,42	0	3,00	3,63	3	8,50	6,58	6,83	5,96	7,83	8,67	11,56	7,83	5,78	6,83	3,78	3,61	9,78	10,00
1980	6,00	5,08	1,26	4,39	1,44	9,17	6,29	0	5,83	4,83	0	3,00	3,08	2	6,17	3,63	5,83	5,42	6,00	5,17	9,11	8,72	3,78	2,39	3,33	2,61	7,78	5,22
1976	6,56	6,58	0,98	5,53	1,23	9,67	8,04	0	7,67	7,42	0	2,33	2,71	2	7,83	6,67	5,83	5,58	6,00	5,92	8,22	6,89	5,78	6,56	4,22	4,39	8,00	6,39
1972	5,89	5,03	1,00	3,92	1,28	8,83	6,08	1	7,50	5,88	0	1,33	1,46	6	6,50	4,33	6,17	5,58	5,00	3,50	8,00	5,89	4,22	3,17	3,56	2,50	7,78	6,33
1969	7,28	7,14	1,19	6,31	1,32	9,67	8,00	0	8,00	7,04	0	4,17	5,13	0	7,17	5,92	7,50	7,71	7,17	6,54	9,78	8,17	5,56	6,39	4,44	4,39	9,33	7,94
1965	8,28	7,61	1,15	6,19	1,42	11,17	8,25	0	9,33	7,96	0	4,33	4,50	0	8,83	6,46	8,50	6,29	7,50	7,96	10,44	9,06	6,89	5,00	5,33	3,94	10,44	9,61
1961	8,00	7,86	1,11	6,33	1,37	9,33	8,13	0	9,50	7,83	0	5,17	5,33	0	9,33	6,67	7,33	6,21	7,33	8,42	11,11	8,61	6,00	5,50	5,56	6,00	9,33	8,28
1957	9,94	8,58	1,14	7,25	1,42	11,83	8,63	0	10,50	8,88	0	7,50	6,25	1	13,33	8,58	7,83	7,38	8,67	7,79	16,00	9,50	8,00	7,61	6,89	6,83	8,89	7,72
1953	13,00	9,36	1,47	7,08	1,84	15,83	8,96	0	12,50	9,00	0	10,67	6,71	0	19,33	9,00	9,83	8,63	9,83	7,04	18,00	10,72	9,11	5,56	9,78	7,50	15,11	9,11
1949	10,94	6,83	1,48	6,03	1,74	16,67	7,96	1	10,67	6,67	1	5,50	4,67	1	17,83	8,33	6,83	4,13	8,17	6,83	14,44	8,50	8,89	5,22	6,00	3,22	14,44	8,78
1919	25,56	29,81	0,92	29,00	0,96	34,83	30,54	0	24,67	31,00	0	17,17	26,67	0	31,00	26,58	21,67	30,13	24,00	31,50	33,56	27,89	22,67	32,00	21,11	24,33	24,89	33,39
Sum								2			1			18														
Average	10,20	9,64	1,13	8,34	1,36	13,74	10,40		10,88	9,87		5,99	6,71		12,36	8,93	8,99	8,90	9,26	9,14	13,87	10,51	7,76	8,18	6,68	6,50	12,49	10,77

Party relaxation

	Goodness	Selection				Initialization									Bound vector						Matrix method							
	Whole data set	Representation		ρ-effect		No			Quota ratio			Apportionment			EL-EL		MF-MF		SD-HA		$CP_{0.01}$		DM		MF		HA	
Year	Initial	Iterations	Ratio	Iterations	Ratio	Goodness	Iterations	Sol.	Goodness	Iterations	Sol.	Goodness	Iterations	Sol.	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations
1994	14,72	7,39	2,27	3,89	3,91	20,67	7,17	0	19,00	6,67	0	4,50	3,08	0	16,67	5,58	13,67	4,38	13,83	6,96	22,44	5,61	11,33	5,83	6,22	2,83	18,89	8,28
1990	19,56	8,56	2,22	6,11	3,23	35,00	7,67	0	18,67	8,38	0	5,00	5,96	0	28,83	8,17	14,83	7,33	15,00	6,50	34,22	7,39	14,67	6,28	10,89	7,89	18,44	7,78
1987	6,72	2,81	2,55	2,11	3,17	9,50	2,79	0	8,50	2,83	0	2,17	1,75	1	6,00	1,54	6,50	3,21	7,67	2,63	9,11	3,00	4,67	2,28	2,22	1,22	10,89	3,33
1983	8,67	4,06	2,27	2,81	3,04	11,50	3,96	0	10,83	3,92	1	3,67	2,42	1	8,17	3,29	8,50	2,83	9,33	4,17	14,00	3,56	6,22	4,06	2,67	1,28	11,78	4,83
1980	6,56	2,83	2,43	1,89	3,32	10,67	2,79	0	6,33	2,54	0	2,67	1,75	1	7,83	2,17	6,33	2,63	5,50	2,29	11,11	3,11	3,56	1,61	4,22	2,50	7,33	2,22
1976	4,61	1,56	3,48	1,42	3,64	5,33	1,67	0	5,33	1,29	1	3,17	1,50	2	5,17	1,42	4,67	1,67	4,00	1,38	6,00	1,44	4,22	1,22	2,89	1,67	5,33	1,61
1972	4,22	1,89	3,18	1,56	3,22	5,50	1,92	1	5,83	1,83	0	1,33	1,42	5	5,00	1,67	3,67	2,58	4,00	0,92	6,67	1,33	2,89	1,00	2,00	1,11	5,33	3,44
1969	11,94	3,44	3,76	2,67	4,56	22,50	3,92	0	10,17	3,33	0	3,17	1,92	0	19,67	2,29	7,50	2,96	8,67	3,92	17,11	2,67	9,78	3,22	7,78	2,78	13,11	3,56
1965	8,61	2,94	3,02	2,39	3,36	14,17	3,17	0	8,50	2,75	0	3,17	2,08	1	11,17	2,50	7,83	2,67	6,83	2,83	14,44	2,89	5,56	2,22	4,89	1,78	9,56	3,78
1961	8,56	3,39	2,76	2,47	3,50	14,83	3,75	0	7,00	2,71	0	3,83	2,33	0	13,33	2,79	6,67	2,50	5,67	3,50	11,78	3,44	6,44	2,22	6,67	3,33	9,33	2,72
1957	13,50	6,44	2,84	2,97	4,12	25,67	5,46	0	11,67	6,71	1	3,17	1,96	1	22,33	3,25	9,50	5,29	8,67	5,58	20,44	4,78	11,78	6,61	7,56	1,89	14,22	5,56
1953	21,11	13,69	1,50	8,50	2,24	38,67	13,96	0	19,00	11,04	0	5,67	8,29	0	31,50	13,83	15,67	10,67	16,17	8,79	32,89	12,94	15,78	7,83	13,78	11,39	22,00	12,22
1949	10,94	6,97	1,50	5,53	1,88	17,83	7,96	0	10,83	6,88	0	4,17	3,92	3	17,83	9,54	7,17	3,38	7,83	5,83	16,67	8,50	8,44	4,83	6,89	3,56	11,78	8,11
1919	25,94	11,44	2,35	9,33	2,74	35,33	12,58	0	30,83	11,50	0	11,67	7,08	0	27,33	9,96	23,83	9,79	26,67	11,42	42,89	11,67	20,22	9,61	14,44	8,78	26,22	11,50
Sum								1			3			15														
Average	11,83	5,53	2,58	3,83	3,28	19,08	5,63		12,32	5,17		4,10	3,25		15,77	4,86	9,74	4,42	9,99	4,77	18,56	5,17	8,97	4,20	6,65	3,72	13,16	5,64

ICELAND (Algorithm data)

Constituency relaxation

	Goodness	Selection				Initialization									Bound vector						Matrix method							
	Whole data set	Representation		ρ-effect		No			Quota ratio			Apportionment			EL-EL		MF-MF		SD-HA		$CP_{0,01}$		DM		MF		HA	
Year	Initial	Iterations	Ratio	Iterations	Ratio	Goodness	Iterations	Sol.	Goodness	Iterations	Sol.	Goodness	Iterations	Sol.	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations
1995	9,39	6,33	1,43	5,69	1,54	14,83	8,29	0	10,00	6,50	0	3,33	3,25	0	11,50	5,17	8,50	5,63	8,17	7,25	12,22	8,22	4,67	4,00	6,22	4,89	14,44	6,94
1991	10,28	7,19	1,44	6,06	1,75	15,83	8,54	0	9,83	6,58	0	5,17	4,75	0	12,67	6,21	9,33	7,33	8,83	6,33	12,22	9,56	6,00	3,72	6,89	4,67	16,00	8,56
1987	11,56	7,89	1,42	7,22	1,57	17,83	8,88	0	13,00	7,67	0	3,83	6,13	0	12,83	9,13	11,17	7,38	10,67	6,17	15,56	9,06	7,11	7,44	7,33	6,72	16,22	7,00
1983	8,33	6,39	1,40	5,44	1,58	14,33	7,75	0	7,33	5,92	1	3,33	4,08	2	11,67	9,71	6,00	3,08	7,33	4,96	10,00	6,72	4,89	7,78	5,33	3,17	13,11	6,00
1979	6,94	5,17	1,34	4,08	1,57	13,33	6,83	1	5,83	4,63	1	1,67	2,42	3	9,50	3,13	6,00	7,42	5,33	3,33	7,33	5,67	4,44	2,67	4,44	3,00	11,56	7,17
1978	7,61	5,14	1,39	4,50	1,63	14,00	7,58	0	6,33	4,79	0	2,50	2,08	1	11,00	4,88	5,67	3,96	6,17	5,63	7,56	5,61	6,44	4,00	5,11	4,39	11,33	5,28
1974	8,00	5,44	1,41	5,00	1,57	14,00	7,13	0	5,50	4,21	0	4,50	4,33	1	11,17	5,54	6,17	5,46	6,67	4,67	8,44	5,50	4,44	2,56	6,44	5,78	12,67	7,06
1971	7,67	7,50	0,93	6,50	1,13	13,00	9,17	0	6,67	7,29	0	3,33	4,54	1	10,17	7,25	6,17	6,58	6,67	7,17	9,78	10,39	5,11	6,11	5,78	4,83	10,00	6,67
1967	7,28	5,61	1,38	4,58	1,62	12,67	6,75	0	6,67	5,29	0	2,50	3,25	3	8,50	2,54	7,17	6,92	6,17	5,83	8,67	8,17	5,33	4,28	5,33	3,11	9,78	4,83
1963	7,78	5,64	1,41	4,42	1,73	13,33	7,00	0	7,50	4,63	0	2,50	3,46	1	9,50	5,38	6,83	5,13	7,00	4,58	9,11	5,78	6,22	6,11	4,89	3,39	10,89	4,83
1959	8,00	5,00	1,55	4,33	1,85	13,50	6,29	0	8,17	5,42	0	2,33	2,29	2	8,83	6,25	7,50	4,25	7,67	3,50	10,22	5,28	6,22	4,39	6,44	5,39	9,11	3,61
Sum								1			2			14														
Average	8,44	6,12	1,37	5,26	1,59	14,24	7,66		7,89	5,72		3,18	3,69		10,67	5,93	7,32	5,74	7,33	5,40	10,10	7,27	5,53	4,82	5,84	4,49	12,28	6,18

Party relaxation

	Goodness	Selection				Initialization									Bound vector						Matrix method							
	Whole data set	Representation		ρ-effect		No			Quota ratio			Apportionment			EL-EL		MF-MF		SD-HA		$CP_{0,01}$		DM		MF		HA	
Year	Initial	Iterations	Ratio	Iterations	Ratio	Goodness	Iterations	Sol.	Goodness	Iterations	Sol.	Goodness	Iterations	Sol.	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations
1995	8,44	5,11	1,69	4,36	1,92	9,67	4,83	0	9,67	5,33	0	6,00	4,04	1	11,17	5,42	7,00	3,46	7,17	5,33	12,67	6,06	3,78	3,44	5,11	3,72	12,22	5,72
1991	8,78	5,94	1,77	4,61	2,15	10,17	6,00	0	9,33	5,54	0	6,83	4,29	0	11,50	4,50	7,17	6,33	7,67	5,00	11,78	5,22	6,22	4,00	7,11	5,17	10,00	6,72
1987	8,56	7,75	1,12	6,39	1,41	9,00	7,71	0	8,67	6,88	0	8,00	6,63	0	10,00	7,33	7,50	6,83	8,17	7,04	10,89	8,28	5,78	5,89	6,00	6,00	11,56	8,11
1983	6,61	5,47	1,26	4,83	1,47	6,50	5,46	0	7,83	5,58	0	5,50	4,42	1	10,17	8,13	4,00	3,08	5,67	4,25	9,56	5,33	4,67	7,00	4,00	2,94	8,22	5,33
1979	3,11	3,25	1,39	2,53	1,59	2,50	3,08	2	3,00	2,50	1	3,83	3,08	1	3,67	1,79	3,00	4,88	2,67	2,00	3,11	3,00	2,00	2,06	2,22	1,83	5,11	4,67
1978	4,61	3,03	2,13	2,36	2,42	5,50	3,25	0	4,00	2,58	0	4,33	2,25	0	5,33	1,58	3,17	2,79	5,33	3,71	5,78	2,17	3,33	2,39	3,78	3,00	5,56	3,22
1974	6,89	4,78	1,63	3,50	2,13	8,00	4,21	0	8,00	4,79	0	4,67	3,42	1	8,33	4,50	6,00	4,46	6,33	3,46	11,33	5,78	2,89	1,61	3,56	4,06	9,78	5,11
1971	7,17	6,33	1,35	4,67	1,66	8,17	6,08	0	7,67	5,79	0	5,67	4,63	0	9,17	7,42	5,83	3,79	6,50	5,29	10,67	8,33	4,89	4,67	4,22	2,78	8,89	6,22
1967	5,44	3,81	2,11	2,64	2,45	6,33	3,42	1	5,83	3,58	0	4,17	2,67	0	5,33	1,67	6,17	3,83	4,83	4,17	7,56	5,33	4,22	2,44	3,33	1,67	6,67	3,44
1963	6,67	2,86	2,68	2,47	3,12	7,67	3,00	0	7,67	2,67	0	4,67	2,33	0	7,17	2,71	6,17	2,42	6,67	2,88	7,11	3,11	5,78	2,94	4,67	2,28	9,11	2,33
1959	6,44	3,53	2,14	2,83	2,51	7,33	3,96	0	8,17	3,71	0	3,83	1,88	1	7,17	3,54	5,67	2,79	6,50	3,21	9,78	3,78	5,11	2,11	4,44	3,39	6,44	3,44
Sum								3			1			5														
Average	6,61	4,71	1,75	3,74	2,08	7,35	4,64		7,26	4,45		5,23	3,60		8,09	4,42	5,61	4,06	6,14	4,21	9,11	5,13	4,42	3,50	4,40	3,35	8,51	4,94

LUXEMBOURG (Algorithm data)

Constituency relaxation

	Goodness	Selection				Initialization									Bound vector						Matrix method							
	Whole data set	Representation		ρ-effect		No			Quota ratio			Apportionment			EL-EL		MF-MF		SD-HA		$CP_{0,01}$		DM		MF		HA	
Year	Initial	Iterations	Ratio	Iterations	Ratio	Goodness	Iterations	Sol.	Goodness	Iterations	Sol.	Goodness	Iterations	Sol	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations
1959	3,67	2,25	1,37	1,67	1,81	6,83	2,88	0	2,83	2,08	1	1,33	0,92	4	5,67	2,21	2,50	1,79	2,83	1,88	4,22	2,44	2,89	2,00	3,11	1,44	4,44	1,94
1954	4,28	2,47	1,80	1,89	2,23	6,67	3,21	0	4,33	2,21	0	1,83	1,13	2	5,17	2,21	4,17	2,58	3,50	1,75	5,78	1,94	3,56	2,11	3,11	2,06	4,67	2,61
1948/51	3,83	2,31	1,48	1,97	1,75	6,33	3,50	0	3,83	2,33	1	1,33	0,58	5	4,67	1,79	3,33	2,58	3,50	2,04	4,67	1,94	2,44	1,89	3,11	2,22	5,11	2,50
1945	3,83	2,44	1,34	1,86	1,72	6,00	3,04	0	4,00	2,29	2	1,50	1,13	5	4,67	2,00	3,50	1,96	3,33	2,50	4,89	2,61	3,33	2,33	2,89	2,00	4,22	1,67
1934/37	2,72	1,86	1,12	1,58	1,30	5,00	2,75	0	1,50	1,25	5	1,67	1,17	5	2,33	1,21	1,50	1,54	4,33	2,42	2,22	2,22	2,44	1,61	2,67	1,44	3,56	1,61
Sum								0			9			21														
Average	3,67	2,27	1,42	1,79	1,76	6,17	3,08		3,30	2,03		1,53	0,99		4,50	1,88	3,00	2,09	3,50	2,12	4,36	2,23	2,93	1,99	2,98	1,83	4,40	2,07

Party relaxation

	Goodness	Selection				Initialization									Bound vector						Matrix method							
	Whole data set	Representation		ρ-effect		No			Quota ratio			Apportionment			EL-EL		MF-MF		SD-HA		$CP_{0,01}$		DM		MF		HA	
Year	Initial	Iterations	Ratio	Iterations	Ratio	Goodness	Iterations	Sol.	Goodness	Iterations	Sol.	Goodness	Iterations	Sol	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations
1959	2,56	2,22	1,42	1,67	1,75	2,00	2,04	1	3,00	2,08	0	2,67	1,71	0	3,33	2,13	2,33	1,50	2,00	2,21	2,44	1,56	2,67	2,39	2,22	1,56	2,89	2,28
1954	4,06	3,11	1,36	2,42	1,60	5,33	3,38	1	4,33	3,25	1	2,50	1,67	2	5,00	3,33	3,33	2,38	3,83	2,58	6,44	4,17	3,33	2,33	2,67	1,61	3,78	2,94
1948/51	3,39	2,31	1,82	2,00	1,94	3,83	2,13	1	3,50	2,46	0	2,83	1,88	0	3,50	1,71	3,33	2,92	3,33	1,83	4,67	2,06	2,67	1,89	2,89	2,00	3,33	2,67
1945	3,89	3,14	1,55	2,67	1,73	4,83	2,83	1	3,67	3,13	0	3,17	2,75	1	4,67	3,13	3,33	2,25	3,67	3,33	4,44	2,56	3,78	3,22	3,78	3,56	3,56	2,28
1934/37	4,06	4,47	1,07	3,19	1,27	5,83	5,75	1	3,00	3,00	1	3,33	2,75	2	5,17	3,79	1,67	1,92	5,33	5,79	5,56	6,22	4,22	3,28	3,56	2,89	2,89	2,94
Sum								5			2			5														
Average	3,59	3,05	1,44	2,39	1,66	4,36	3,23		3,50	2,78		2,90	2,15		4,33	2,82	2,80	2,19	3,63	3,15	4,71	3,31	3,33	2,62	3,02	2,32	3,29	2,62

NORWAY (Algorithm data)

Constituency relaxation

	Goodness	Selection				Initialization									Bound vector						Matrix method							
	Whole data set	Representation		ρ-effect		No			Quota ratio			Apportionment			EL-EL		MF-MF		SD-HA		$CP_{0,01}$		DM		MF		HA	
Year	Initial	Iterations	Ratio	Iterations	Ratio	Goodness	Iterations	Sol.	Goodness	Iterations	Sol.	Goodness	Iterations	Sol.	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations
1993	17,28	20,42	0,86	19,53	0,96	20,83	21,54	0	16,00	17,38	0	15,00	21,00	0	17,50	16,33	17,00	22,88	17,33	20,71	21,11	25,50	11,33	14,56	12,89	16,11	23,78	23,72
1989	16,28	18,83	0,90	16,78	1,04	20,50	20,54	0	16,17	16,38	0	12,17	16,50	0	16,83	18,00	15,50	19,79	16,50	15,63	19,11	25,67	11,11	15,50	11,78	14,33	23,11	15,72
1985	18,11	17,94	1,00	15,72	1,15	20,00	17,71	0	14,33	14,79	0	20,00	18,00	0	18,83	15,29	17,50	18,42	18,00	16,79	25,56	18,61	11,11	13,83	12,44	15,22	23,33	19,67
1981	18,00	17,81	1,05	15,44	1,20	22,17	18,79	0	13,83	15,25	0	18,00	15,83	0	18,83	12,38	16,83	18,46	18,33	19,04	20,22	16,72	14,44	15,11	15,33	13,78	22,00	20,89
1977	13,89	12,94	1,03	11,11	1,22	17,50	12,79	0	9,83	10,38	0	14,33	12,92	0	16,67	12,29	12,17	11,21	12,83	12,58	19,11	16,61	8,44	8,72	10,00	8,78	18,00	14,00
1973	18,33	19,47	0,97	18,25	1,02	24,67	21,25	0	16,50	18,21	0	13,83	17,13	0	19,33	17,33	16,33	20,25	19,33	19,00	22,00	21,72	12,67	17,50	14,22	16,89	24,44	19,33
1969	16,50	18,08	0,92	16,83	1,01	23,17	19,58	0	15,33	17,75	0	11,00	15,04	0	17,67	17,54	16,00	18,38	15,83	16,46	20,89	20,50	12,44	19,33	10,00	11,39	22,67	18,61
1965	16,89	22,58	0,82	21,11	0,85	23,00	23,92	0	15,67	20,58	0	12,00	21,04	1	17,50	15,13	16,00	25,08	17,17	25,33	19,78	30,78	14,00	15,94	14,00	16,33	19,78	24,33
1961	17,61	19,00	0,94	17,69	1,03	24,00	20,00	0	15,50	17,25	0	13,33	17,79	0	23,33	16,58	14,50	18,96	15,00	19,50	22,67	23,22	15,78	16,78	11,11	12,78	20,89	20,61
1957	16,94	17,19	0,98	15,78	1,07	23,33	17,54	0	14,83	16,04	0	12,67	15,88	0	21,17	16,67	15,17	19,54	14,50	13,25	20,67	19,72	15,33	16,00	11,78	12,61	20,00	17,61
1953	17,89	19,14	0,94	18,08	1,03	25,33	20,71	0	15,33	18,46	0	13,00	16,67	0	20,83	17,92	17,33	20,13	15,50	17,79	23,11	25,00	14,89	18,78	12,22	11,78	21,33	18,89
1949	22,22	22,06	0,98	21,00	1,03	31,50	24,58	0	19,17	21,25	0	16,00	18,75	1	24,83	18,71	20,00	19,50	21,83	26,38	27,56	29,28	17,11	18,22	17,56	18,33	26,67	20,28
Sum								0			0			2														
Average	17,50	18,79	0,95	17,28	1,05	23,00	19,91		15,21	16,98		14,28	17,21		19,44	16,18	16,19	19,38	16,85	18,54	21,82	22,78	13,22	15,86	12,78	14,03	22,17	19,47

Party relaxation

	Goodness	Selection				Initialization									Bound vector						Matrix method							
	Whole data set	Representation		ρ-effect		No			Quota ratio			Apportionment			EL-EL		MF-MF		SD-HA		$CP_{0,01}$		DM		MF		HA	
Year	Initial	Iterations	Ratio	Iterations	Ratio	Goodness	Iterations	Sol.	Goodness	Iterations	Sol.	Goodness	Iterations	Sol.	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations
1993	24,94	12,53	1,90	11,03	2,23	33,67	13,33	0	33,33	14,21	0	7,83	7,79	0	24,50	9,04	25,17	13,38	25,17	12,92	45,78	15,22	14,22	7,22	10,67	10,78	29,11	13,89
1989	19,50	11,00	1,76	8,31	2,31	25,67	11,08	0	25,83	10,42	0	7,00	7,46	0	20,83	10,25	18,83	10,38	18,83	8,33	33,78	14,61	11,56	8,78	7,78	7,78	24,89	7,44
1985	27,11	8,78	2,91	7,58	3,36	38,67	9,88	0	37,00	9,25	0	5,67	5,42	0	28,00	8,04	26,33	8,92	27,00	7,58	54,89	8,67	13,11	5,78	10,89	8,11	29,56	10,17
1981	26,83	11,31	2,47	8,00	3,42	36,50	11,88	0	37,33	12,29	0	6,67	4,79	0	29,33	7,83	25,00	10,33	26,17	10,79	48,22	10,89	16,00	8,89	14,22	7,39	28,89	11,44
1977	22,00	9,28	1,95	6,97	2,69	33,17	11,25	0	27,83	9,29	0	5,00	3,83	0	25,17	8,54	20,00	6,96	20,83	8,88	48,67	11,39	10,67	6,56	10,00	5,89	18,67	8,67
1973	21,22	12,17	1,71	10,53	1,94	29,17	13,04	0	26,50	13,71	0	8,00	7,29	0	24,00	9,71	19,83	12,88	19,83	11,46	37,33	16,11	14,00	10,06	12,44	9,89	21,11	9,33
1969	18,33	9,67	2,32	8,08	2,64	24,83	10,04	0	23,50	9,71	0	6,67	6,88	0	19,50	6,17	17,83	10,33	17,67	10,13	33,56	8,94	11,11	8,39	8,00	7,44	20,67	10,72
1965	21,39	10,22	2,25	9,42	2,50	28,33	11,50	0	27,00	10,92	0	8,83	7,04	0	20,00	5,67	21,83	11,42	22,33	12,38	37,56	12,83	14,67	6,28	10,67	8,33	22,67	11,83
1961	19,56	9,92	2,06	8,31	2,34	26,67	9,96	0	25,00	10,79	0	7,00	6,58	0	21,33	7,46	18,50	9,71	18,83	10,17	35,33	12,17	14,00	9,11	7,56	5,67	21,33	9,50
1957	21,11	7,33	2,65	6,72	2,94	29,17	7,54	0	27,33	7,96	0	6,83	5,58	0	25,83	7,50	18,33	7,25	19,17	6,33	37,56	9,06	14,89	6,50	9,56	4,78	22,44	7,78
1953	20,89	7,67	2,66	6,72	3,18	27,33	7,75	0	27,17	7,75	0	8,17	6,08	0	23,17	6,75	18,50	7,25	21,00	7,58	35,33	8,67	16,22	7,39	8,22	5,89	23,78	6,83
1949	20,44	8,36	2,58	7,50	2,98	28,17	8,54	1	25,83	9,00	0	7,33	6,25	0	24,50	5,13	17,33	8,79	19,50	9,88	36,89	8,00	13,78	8,17	10,22	7,22	20,89	8,33
Sum								1			0			0														
Average	21,94	9,85	2,27	8,26	2,71	30,11	10,48		28,64	10,44		7,08	6,25		23,85	7,67	20,62	9,80	21,36	9,70	40,41	11,38	13,69	7,76	10,02	7,43	23,67	9,66

SWEDEN (Algorithm data)

Constituency relaxation

	Goodness	Selection				Initialization								Bound vector						Matrix method							
	Whole data set	Representation		ρ-effect		No			Quota ratio			Apportionment			EL-EL		MF-MF		SD-HA		$CP_{0,01}$		DM		MF		HA	
Year	Initial	Iterations	Ratio	Iterations	Ratio	Goodness	Iterations	Sol.	Goodness	Iterations	Sol.	Goodness	Iterations	Sol.	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations
1994	27,44	22,86	1,18	21,61	1,26	32,50	24,46	0	32,50	24,00	0	17,33	18,25	0	27,00	21,13	27,67	22,96	27,67	22,63	37,56	28,89	19,78	18,72	13,11	18,83	39,33	22,50
1991	22,89	24,22	0,93	22,14	1,03	27,17	26,08	0	25,17	23,29	0	16,33	20,17	0	18,17	18,00	24,33	24,29	26,17	27,25	31,11	29,78	16,00	19,00	15,33	19,00	29,11	24,94
1988	24,94	25,83	1,00	23,28	1,11	29,33	25,17	0	27,00	24,54	0	18,50	23,96	0	22,33	21,13	26,33	30,17	26,17	22,38	32,67	28,06	22,67	24,28	14,22	16,56	30,22	29,33
1985	15,22	15,50	1,01	16,00	1,05	18,00	17,96	0	15,67	16,08	0	12,00	13,21	0	13,17	15,04	16,17	16,29	16,33	15,92	22,22	23,67	10,22	12,22	7,78	8,17	20,67	18,94
1982	20,89	21,81	0,99	19,00	1,14	22,83	20,13	0	22,50	21,08	0	17,33	20,00	0	17,00	14,79	22,83	22,58	22,83	23,83	27,78	18,67	17,11	21,17	14,22	18,22	24,44	23,56
1979	16,89	19,53	0,89	18,61	0,97	19,67	21,96	0	19,17	18,92	0	11,83	16,33	0	14,00	12,17	18,00	22,54	18,67	22,50	23,11	20,17	11,56	18,44	9,11	17,78	23,78	19,89
1976	19,50	23,19	0,86	22,92	0,87	23,83	24,67	0	21,83	24,54	0	12,83	19,96	0	16,00	19,96	20,83	22,54	21,67	26,67	26,00	30,39	13,56	21,67	11,33	16,67	27,11	23,50
1973	19,67	21,83	0,90	19,33	1,02	22,83	23,58	0	21,83	21,54	0	14,33	16,63	0	16,83	18,08	20,00	22,08	22,17	21,58	26,22	22,78	14,44	18,89	11,56	16,83	26,44	23,83
1970	17,89	18,53	0,99	16,67	1,18	22,00	20,29	0	19,00	16,96	0	12,67	15,54	0	14,83	14,75	18,50	18,71	20,33	19,33	25,56	18,28	11,56	11,39	10,22	10,89	24,22	29,83
1968	16,50	17,67	0,99	16,31	1,08	18,50	19,17	0	17,33	16,92	0	13,67	14,88	0	17,33	16,71	16,50	14,92	15,67	19,33	21,78	19,33	13,11	19,17	10,89	16,28	20,22	13,17
1964	15,56	19,67	0,81	17,75	0,90	17,17	20,21	0	16,33	19,42	0	13,17	16,50	0	14,83	19,17	16,00	19,00	15,83	17,96	20,00	20,11	11,33	18,44	8,89	17,17	22,00	19,11
1960	13,17	14,89	0,93	14,19	1,04	16,00	15,75	0	14,33	15,75	0	9,17	12,13	0	13,67	13,83	12,83	14,21	13,00	15,58	17,33	16,33	8,44	10,89	9,56	13,00	17,33	17,94
1958	12,89	17,50	0,77	16,53	0,86	16,00	19,50	0	14,50	17,46	0	8,17	14,08	0	13,83	16,58	13,50	19,08	11,33	15,38	16,44	18,78	10,44	13,44	9,33	12,39	15,33	23,44
1956	18,00	20,67	0,88	19,69	0,95	21,83	23,71	0	18,50	20,17	0	13,67	16,67	0	17,33	17,50	18,17	24,17	18,50	18,88	22,00	25,22	14,67	19,17	13,11	17,39	22,22	18,94
1952	18,11	20,78	0,89	19,86	0,98	21,83	22,92	0	19,83	20,21	0	12,67	17,83	0	16,17	18,08	18,50	22,67	19,67	20,21	24,67	22,28	15,11	22,78	14,44	20,39	18,22	15,83
1948	17,61	18,81	0,94	18,03	1,01	21,33	20,71	0	19,33	18,17	0	12,17	16,38	0	16,50	17,25	17,83	17,17	18,50	20,83	22,22	24,17	12,89	13,56	12,44	16,28	22,89	19,67
1944	18,89	21,44	0,89	20,08	0,96	23,83	22,13	0	21,83	22,33	0	11,00	17,83	0	18,83	18,54	18,50	22,46	19,33	21,29	25,78	20,61	15,11	16,56	12,67	20,06	22,00	25,83
1940	17,56	20,89	0,85	18,25	0,95	24,00	23,29	0	17,17	18,79	0	11,50	16,63	0	17,83	16,96	17,50	21,92	17,33	19,83	23,78	21,94	13,78	16,72	13,33	19,72	19,33	19,89
1936	16,44	19,33	0,89	18,56	0,92	19,00	21,96	0	15,33	18,46	0	15,00	16,42	0	17,83	17,17	15,17	19,79	16,33	19,88	24,00	26,56	10,44	11,78	10,67	11,67	20,67	25,78
1932	18,11	23,75	0,77	22,67	0,81	20,50	23,92	0	17,83	22,67	0	16,00	23,04	0	17,50	22,21	19,50	24,63	17,33	22,79	23,78	27,83	13,56	19,28	13,78	23,50	21,33	22,22
1928	16,33	18,00	0,92	17,56	0,95	21,17	20,58	0	17,17	18,75	0	10,67	14,00	0	16,00	16,83	16,67	18,17	16,33	18,33	23,56	22,39	11,33	13,33	11,56	14,33	18,89	21,06
1924	20,06	24,36	0,85	21,06	0,98	24,00	27,08	0	16,83	22,29	0	19,33	18,75	0	19,67	18,21	19,33	24,83	21,17	25,08	30,67	27,78	14,89	25,11	13,56	14,22	21,11	23,72
1921	15,56	19,03	0,82	18,47	0,88	20,17	21,00	0	14,50	18,04	0	12,00	17,21	0	18,50	21,79	14,00	18,29	14,17	16,17	20,00	25,61	11,33	16,06	11,11	13,39	19,78	19,94
1920	31,17	54,67	0,65	58,53	0,64	40,67	62,54	0	33,67	55,92	0	19,17	51,33	0	32,17	59,63	31,17	48,08	30,17	62,08	39,33	112,11	23,56	30,33	24,22	40,78	37,56	43,17
1917	28,06	40,58	0,71	39,39	0,75	33,83	43,67	0	29,67	37,63	0	20,67	38,67	0	30,00	39,88	26,83	41,54	27,33	38,54	40,22	68,22	18,22	25,33	20,00	28,61	33,78	37,78
1914 II	17,72	22,81	0,79	21,75	0,87	28,00	32,33	0	20,67	26,75	0	4,50	7,75	0	19,33	22,71	17,17	21,38	16,67	22,75	19,11	25,06	16,44	23,61	14,00	15,39	21,33	25,06
1914 I	17,39	21,42	0,93	20,61	0,98	27,17	31,92	0	21,50	26,13	0	3,50	5,00	0	19,50	20,13	15,67	18,96	17,00	23,96	18,22	19,33	14,00	19,00	15,78	17,06	21,56	28,67
1911	17,94	23,56	0,73	23,83	0,73	28,83	34,00	0	19,00	26,29	0	6,00	10,79	0	21,50	23,54	15,83	23,75	16,50	23,79	20,22	23,28	15,78	23,56	14,00	22,17	21,78	25,78
Sum								0			0			0														
Average	19,01	22,61	0,88	21,52	0,96	23,64	25,38		20,36	22,61		13,04	18,21		18,49	20,42	19,12	22,76	19,43	23,03	25,19	28,13	14,33	18,71	12,87	17,74	23,67	23,69

SWEDEN (Algorithm data)

Party relaxation

	Goodness	Selection				Initialization									Bound vector						Matrix method							
	Whole data set	Representation		ρ-effect		No			Quota ratio			Apportionment			EL-EL		MF-MF		SD-HA		$CP_{0,01}$		DM		MF		HA	
Year	Initial	Iterations	Ratio	Iterations	Ratio	Goodness	Iterations	Sol.	Goodness	Iterations	Sol.	Goodness	Iterations	Sol.	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations	Goodness	Iterations
1994	41,28	11,97	3,11	8,58	4,26	57,00	11,50	0	55,67	12,04	0	11,17	7,29	0	37,50	9,04	43,33	10,79	43,00	11,00	74,44	15,00	24,89	8,89	9,11	7,44	56,67	9,78
1991	36,56	11,67	2,99	9,03	3,71	51,50	11,92	0	49,00	11,67	0	9,17	7,46	0	34,50	8,08	36,33	11,58	38,83	11,38	64,22	16,00	22,67	9,00	10,44	7,61	48,89	8,78
1988	34,61	10,25	3,13	9,69	3,41	48,17	11,50	0	47,17	11,00	0	8,50	7,42	0	33,17	9,88	35,67	11,83	35,00	8,21	59,78	12,22	21,33	9,33	11,56	7,33	45,78	11,00
1985	24,50	6,03	3,50	5,42	3,95	33,67	6,33	0	33,33	6,25	1	6,50	4,58	0	23,50	4,71	24,83	6,75	25,17	5,71	52,00	7,78	11,56	5,72	4,89	2,56	29,56	6,83
1982	29,72	8,22	3,08	6,06	4,54	42,00	8,75	0	41,33	8,08	0	5,83	4,58	0	28,50	6,29	29,83	6,33	30,83	8,79	58,67	8,11	17,56	6,22	8,00	5,44	34,67	8,78
1979	24,94	8,28	2,71	6,81	3,56	34,83	9,21	0	34,33	8,50	0	5,67	4,92	0	22,67	5,54	26,00	8,54	26,17	8,54	50,44	9,22	11,78	8,44	8,44	5,28	29,11	7,22
1976	26,89	9,72	2,73	7,86	3,36	37,17	9,71	0	36,50	9,71	0	7,00	6,96	0	23,17	8,96	28,83	8,17	28,67	9,25	52,89	13,50	11,78	8,22	9,11	5,06	33,78	8,39
1973	28,89	8,50	2,94	6,25	4,46	40,50	8,46	0	39,17	8,58	0	7,00	5,08	0	26,67	6,79	30,17	8,58	29,83	6,75	55,78	7,72	15,56	7,44	9,56	5,89	34,67	8,44
1970	27,17	8,08	3,18	6,47	4,45	38,17	8,83	0	37,00	7,79	0	6,33	5,21	0	25,33	7,88	27,50	7,29	28,67	6,67	54,00	6,72	11,78	6,67	10,00	4,56	32,89	11,17
1968	29,17	8,00	3,24	6,39	4,01	41,00	8,83	0	40,00	8,54	0	6,50	4,21	0	31,00	6,54	28,17	7,83	28,33	7,21	61,33	10,17	17,11	7,50	10,67	5,50	27,56	5,61
1964	24,78	9,19	2,53	6,75	3,47	34,83	9,21	0	33,67	9,71	0	5,83	5,00	0	26,00	6,42	24,17	9,29	24,17	8,21	47,56	9,39	9,78	8,56	11,33	6,11	30,44	7,83
1960	18,28	5,50	3,11	4,67	3,83	26,00	6,21	0	25,00	6,17	0	3,83	2,88	1	20,00	4,08	16,67	4,79	18,17	6,38	33,78	5,89	10,00	4,22	6,44	5,00	22,89	5,22
1958	15,33	5,47	2,69	4,86	2,95	21,17	6,54	0	19,17	5,33	0	5,67	3,63	0	16,00	6,13	16,00	5,54	14,00	3,83	23,56	6,50	9,78	4,06	5,56	4,39	22,44	5,72
1956	20,67	6,72	2,89	5,42	3,63	28,67	6,92	0	26,33	6,88	0	7,00	4,42	0	21,17	6,25	20,17	6,42	20,67	5,54	36,89	8,39	12,89	4,83	9,56	5,83	23,33	5,22
1952	21,78	5,61	3,97	5,39	4,14	29,83	6,46	0	29,00	6,38	0	6,50	3,67	0	22,17	5,83	21,50	5,79	21,67	4,88	39,11	4,83	14,22	7,33	11,56	4,83	22,22	5,00
1948	20,17	5,28	3,40	4,31	3,99	28,50	5,63	0	27,33	5,58	0	4,67	3,17	3	21,67	4,29	19,17	4,71	19,67	5,38	36,67	6,83	13,11	4,06	8,00	3,67	22,89	4,61
1944	18,06	5,89	2,96	4,64	3,93	25,67	5,75	0	22,00	6,08	0	6,50	3,96	0	21,33	4,96	16,00	4,63	16,83	6,21	28,89	5,67	10,22	3,83	9,33	5,00	23,78	6,56
1940	21,89	7,14	3,05	5,42	3,63	30,67	7,54	0	28,83	6,58	0	6,17	4,71	0	23,67	4,92	21,33	8,21	20,67	5,71	43,11	8,89	11,11	5,89	9,11	4,67	24,22	5,67
1936	24,33	7,97	2,76	7,11	3,15	33,33	8,08	0	33,17	8,21	0	6,50	6,33	0	27,00	6,50	22,50	8,04	23,50	8,08	54,44	11,06	8,89	5,44	9,11	4,94	24,89	8,72
1932	22,94	9,58	2,24	6,00	3,77	32,83	8,92	0	28,50	9,88	0	7,50	4,58	0	25,33	7,54	22,83	7,96	20,67	7,88	42,67	10,39	13,11	6,17	12,44	7,33	23,56	7,28
1928	21,44	7,22	2,79	4,92	4,06	29,83	7,29	0	28,00	6,96	0	6,50	3,96	0	24,33	5,92	20,50	5,92	19,50	6,38	44,00	9,50	10,22	4,22	9,33	4,39	22,22	6,17
1924	30,17	9,19	2,87	7,33	3,59	41,50	10,25	0	40,83	9,63	0	8,17	4,92	0	32,33	8,08	28,33	8,33	29,83	8,38	66,67	10,67	14,67	8,22	12,67	5,50	26,67	8,67
1921	18,78	7,72	2,45	5,61	3,40	25,67	7,92	0	23,33	7,83	0	7,33	4,25	0	23,67	7,25	16,17	7,00	16,50	5,75	35,78	10,39	12,44	4,67	8,44	4,61	18,44	7,00
1920	24,17	10,69	2,38	8,08	3,15	31,50	8,88	0	28,67	10,33	0	12,33	8,96	0	25,83	9,33	23,67	8,92	23,00	9,92	42,67	12,56	14,67	6,67	17,56	9,28	21,78	9,06
1917	24,61	6,28	4,08	5,42	4,50	32,33	6,63	0	29,50	7,08	0	12,00	3,83	0	28,00	6,17	22,83	6,08	23,00	5,29	41,78	7,78	14,44	5,00	13,78	5,56	28,44	5,06
1914 II	8,28	2,81	3,57	2,33	4,04	10,67	3,04	0	9,50	2,79	0	4,67	1,88	0	8,00	1,92	7,33	2,46	9,50	3,33	11,56	2,50	7,78	2,89	7,11	2,44	6,67	2,44
1914 I	5,39	3,39	2,46	2,39	2,80	5,83	3,00	0	6,17	3,08	0	4,17	2,58	0	5,83	2,08	4,00	3,92	6,33	2,67	5,78	3,67	3,56	2,67	4,67	2,00	7,56	3,22
1911	12,61	3,33	4,65	2,72	5,05	16,00	3,42	0	14,17	3,33	0	7,67	2,33	0	15,00	2,88	10,83	3,58	12,00	2,63	19,33	3,00	9,78	2,11	8,89	3,00	12,44	4,00
Sum								0			1			4														
Average	23,48	7,49	3,05	5,93	3,81	32,46	7,74		30,95	7,64		7,02	4,74		24,05	6,22	23,02	7,12	23,36	6,78	44,21	8,73	13,10	6,01	9,52	5,19	27,09	6,91

Description of Algorithm tests tables

The data in the Algorithm test tables have been calculated from figures in the Algorithm data tables. We use the following abbreviations:

N = No initialization
Q = Quota ratio initialization
A = Apportionment initialization

C = Constituency relaxation
P = Party relaxation

R = Representation selection
O = ρ-effect selection

Initializations are tested against each other based on iterations and initial measure of goodness. The difference X - Y, where X stands for one initialization and Y for another, is calculated as:

[(Average value for X with C + Average value for X with P) -
(Average value for Y with C + Average value for Y with P)] / 2

Relaxations are tested based on iterations, the average decrease in the measure of goodness per iteration, termed "Ratio" in the tables, and the initial measure of goodness. The ratio P/C is calculated as:

(Average value with R and P + Average value with O and P) /
(Average value with R and C + Average value with O and C)

The difference P - C (for ρ) is calculated as:

Average initial measure of goodness with P - Average initial measure of goodness with C

Selections are tested based on iterations and the average decrease in the measure of goodness per iteration. The ratio O/R is calculated as:

(Average value with O and C + Average value with O and P) /
(Average value with R and C + Average value with R and P)

NB. P-values for Initializations and Selections are based on one-sided tests, while P-values for Relaxations are based on two-sided tests.

Algorithm tests

Austria	Initializations				Relaxations			Selections	
	Iterations		Initial ρ		Iterations	"Ratio"	ρ	Iterations	"Ratio"
	Differences		Differences		Ratio	Ratio	Difference	Ratio	Ratio
Year	N - Q	Q - A	N - Q	Q - A	P/C	P/C	P - C	O/R	O/R
1994	-0,27	2,50	1,17	6,50	0,55	1,62	-1,56	0,79	1,29
1990	0,48	2,63	2,08	6,34	0,60	2,06	2,00	0,77	1,35
1986	0,23	2,30	0,67	4,25	0,44	2,70	0,27	0,84	1,25
1983	0,81	2,15	1,67	4,83	1,08	2,47	6,28	0,49	1,82
All Austrian sets									
Number of data sets: 4									
Estimated mean	0,31	2,39	1,40	5,48	0,67	2,21	1,75	0,72	1,43
Standard error of the estimate	0,23	0,11	0,31	0,56	0,14	0,24	1,68	0,08	0,13
t-statistic	1,36	22,43	4,56	9,86	2,33	-5,12	1,04	3,52	-3,27
P-value (two-sided for relaxation)	0,1334	0,0001	0,0099	0,0011	0,1021	0,0144	0,3739	0,0195	0,0234

Denmark	Initializations				Relaxations			Selections	
	Iterations		Initial ρ		Iterations	"Ratio"	ρ	Iterations	"Ratio"
	Differences		Differences		Ratio	Ratio	Difference	Ratio	Ratio
Year	N - Q	Q - A	N - Q	Q - A	P/C	P/C	P - C	O/R	O/R
1994	-0,84	2,59	0,34	13,75	0,62	2,12	9,33	0,83	1,26
1990	0,50	3,19	1,09	15,08	0,75	1,80	9,66	0,81	1,27
1988	0,60	2,92	0,59	15,34	0,80	1,50	6,50	0,90	1,16
1987	0,16	3,94	1,25	16,76	0,79	1,67	10,28	0,89	1,13
1984	-0,17	3,38	1,09	14,33	0,75	1,99	10,61	0,91	1,15
1981	0,60	3,01	2,25	12,75	0,75	1,83	7,16	0,84	1,22
1979	-0,36	2,92	1,59	13,76	0,69	1,77	8,89	0,95	1,12
1977	0,15	1,15	0,16	15,67	0,81	1,83	10,11	0,93	1,09
1975	1,23	2,03	1,42	14,75	0,76	1,84	11,06	0,94	1,06
1973	2,50	0,29	1,59	12,00	0,78	1,52	5,56	0,91	1,12
1971	1,02	3,33	3,08	12,92	0,60	2,32	10,05	0,84	1,20
1968	1,92	1,33	4,17	10,25	0,64	2,04	6,44	0,80	1,36
1966	0,61	-0,53	1,92	12,58	0,40	3,13	7,06	0,88	1,17
All Danish sets									
Number of data sets: 13									
Estimated mean	0,61	2,27	1,58	13,84	0,70	1,95	8,67	0,88	1,18
Standard error of the estimate	0,25	0,37	0,31	0,48	0,03	0,12	0,52	0,01	0,02
t-statistic	2,42	6,13	5,14	28,57	9,42	-8,16	16,76	8,86	-7,66
P-value (two-sided for relaxation)	0,0162	0,0000	0,0001	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000

Finland	Initializations				Relaxations			Selections	
	Iterations		Initial ρ		Iterations	"Ratio"	ρ	Iterations	"Ratio"
	Differences		Differences		Ratio	Ratio	Difference	Ratio	Ratio
Year	N - Q	Q - A	N - Q	Q - A	P/C	P/C	P - C	O/R	O/R
1995	-0,92	4,28	2,00	12,25	0,74	2,05	10,45	0,84	1,30
1991	-0,19	2,79	1,59	11,50	0,67	2,31	8,55	0,83	1,28
1987	0,02	5,04	2,08	14,59	0,90	1,84	12,45	0,85	1,39
1983	1,36	2,57	2,17	6,67	0,77	1,94	5,12	0,91	1,14
1979	0,28	0,71	1,67	10,00	0,85	2,42	11,33	0,80	1,38
1975	0,48	2,67	1,59	14,75	0,93	1,95	13,66	0,83	1,34
1972	0,88	1,90	1,59	3,92	0,61	1,93	0,84	0,90	1,12
1970	1,26	0,26	1,92	4,76	0,57	1,89	0,44	0,89	1,13
1966	2,42	1,19	3,17	6,59	0,55	2,18	3,44	0,92	1,12
1962	0,38	2,48	2,34	6,92	0,50	2,67	2,23	0,88	1,21
1958	1,33	2,29	2,42	3,84	0,56	1,96	0,17	0,85	1,22
1954	1,25	2,92	1,75	5,50	0,46	1,98	-0,67	0,95	1,17
1951	0,63	2,36	0,92	5,67	0,49	1,74	-1,89	0,86	1,25
1948	0,94	2,07	2,92	4,92	0,57	1,58	-1,28	0,90	1,21
1945	0,98	1,57	2,92	5,75	0,64	1,88	2,06	0,91	1,17
1939	1,83	1,03	3,00	8,58	0,67	2,05	6,66	0,86	1,27
1936	1,19	3,48	3,00	11,67	0,64	2,00	5,78	0,85	1,22
All Finnish sets									
Number of data sets: 17									
Estimated mean	0,83	2,33	2,18	8,11	0,65	2,02	4,67	0,87	1,23
Standard error of the estimate	0,19	0,30	0,16	0,88	0,03	0,06	1,23	0,01	0,02
t-statistic	4,33	7,85	13,88	9,17	10,11	-16,34	3,78	13,29	-10,84
P-value (two-sided for relaxation)	0,0003	0,0000	0,0000	0,0000	0,0000	0,0000	0,0016	0,0000	0,0000

Algorithm tests

Germany	Initializations				Relaxations			Selections	
	Iterations		Initial ρ		Iterations	"Ratio"	ρ	Iterations	"Ratio"
	Differences		Differences		Ratio	Ratio	Difference	Ratio	Ratio
Year	N - Q	Q - A	N - Q	Q - A	P/C	P/C	P - C	O/R	O/R
1994	1,23	3,57	3,34	11,25	0,47	2,65	1,22	0,75	1,55
1990	-0,28	2,34	9,50	8,42	0,54	2,65	6,06	0,78	1,38
1987	0,06	2,36	0,34	6,25	0,45	2,39	0,05	0,81	1,22
1983	0,40	3,15	1,25	6,67	0,49	2,33	0,95	0,79	1,33
1980	0,86	1,27	3,84	3,25	0,50	2,13	0,56	0,79	1,29
1976	0,50	2,25	1,00	3,75	0,25	3,22	-1,95	0,85	1,09
1972	0,15	2,42	0,50	5,34	0,39	2,81	-1,67	0,79	1,08
1969	0,77	1,66	7,00	5,42	0,45	3,31	4,66	0,85	1,19
1965	0,36	2,07	3,76	5,17	0,39	2,48	0,33	0,81	1,15
1961	0,67	1,44	3,83	3,75	0,41	2,52	0,56	0,78	1,26
1957	-0,75	3,69	7,67	5,75	0,59	2,72	3,56	0,68	1,39
1953	1,44	2,52	11,50	7,58	1,35	1,13	8,11	0,68	1,37
1949	1,19	2,48	6,50	5,92	0,97	1,05	0,00	0,84	1,21
1919	0,31	4,38	7,33	13,33	0,35	2,71	0,38	0,93	1,13
All German sets									
Number of data sets: 14									
Estimated mean	0,49	2,54	4,81	6,56	0,54	2,44	1,63	0,80	1,26
Standard error of the estimate	0,16	0,24	0,94	0,76	0,08	0,17	0,78	0,02	0,04
t-statistic	3,08	10,78	5,13	8,63	6,00	-8,25	2,10	11,58	-7,26
P-value (two-sided for relaxation)	0,0044	0,0000	0,0001	0,0000	0,0000	0,0000	0,0557	0,0000	0,0000
(1949 - 1987)									
Number of data sets: 11									
Estimated mean	0,51	2,30	4,29	5,35	0,57	2,37	1,38	0,79	1,23
Standard error of the estimate	0,18	0,21	1,07	0,40	0,10	0,22	0,89	0,02	0,03
t-statistic	2,87	10,77	4,00	13,38	4,53	-6,25	1,55	11,68	-7,34
P-value (two-sided for relaxation)	0,0083	0,0000	0,0013	0,0000	0,0011	0,0001	0,1533	0,0000	0,0000

Iceland	Initializations				Relaxations			Selections	
	Iterations		Initial ρ		Iterations	"Ratio"	ρ	Iterations	"Ratio"
	Differences		Differences		Ratio	Ratio	Difference	Ratio	Ratio
Year	N - Q	Q - A	N - Q	Q - A	P/C	P/C	P - C	O/R	O/R
1995	0,65	2,27	2,42	5,17	0,79	1,22	-0,95	0,88	1,11
1991	1,21	1,54	3,42	3,58	0,80	1,23	-1,50	0,81	1,21
1987	1,02	0,90	2,58	4,92	0,94	0,85	-3,00	0,87	1,17
1983	0,86	1,50	2,84	3,17	0,87	0,92	-1,72	0,87	1,15
1979	1,39	0,82	3,50	1,67	0,62	1,02	-3,83	0,79	1,16
1978	1,73	1,52	4,59	1,75	0,56	1,51	-3,00	0,84	1,15
1974	1,17	0,63	4,25	2,17	0,79	1,26	-1,11	0,83	1,22
1971	1,09	1,96	3,42	2,67	0,79	1,46	-0,50	0,81	1,22
1967	0,65	1,48	3,25	2,92	0,63	1,52	-1,84	0,77	1,17
1963	1,35	0,76	2,92	4,00	0,53	1,85	-1,11	0,81	1,19
1959	0,56	2,48	2,25	5,09	0,68	1,37	-1,56	0,84	1,18
All Icelandic sets									
Number of data sets: 11									
Estimated mean	1,06	1,44	3,22	3,37	0,73	1,29	-1,83	0,83	1,18
Standard error of the estimate	0,11	0,19	0,22	0,39	0,04	0,09	0,31	0,01	0,01
t-statistic	9,71	7,66	14,62	8,64	6,96	-3,28	-5,92	16,05	-16,73
P-value (two-sided for relaxation)	0,0000	0,0000	0,0000	0,0000	0,0000	0,0083	0,0001	0,0000	0,0000

Algorithm tests

Luxembourg	Initializations				Relaxations			Selections	
	Iterations		Initial ρ		Iterations	"Ratio"	ρ	Iterations	"Ratio"
	Differences		Differences		Ratio	Ratio	Difference	Ratio	Ratio
Year	N - Q	Q - A	N - Q	Q - A	P/C	P/C	P - C	O/R	O/R
1959	0,38	0,77	1,50	0,92	0,99	1,00	-1,11	0,75	1,28
1954	0,57	1,33	1,67	2,17	1,27	0,73	-0,22	0,77	1,21
1948/51	0,42	1,17	1,42	1,59	1,01	1,16	-0,44	0,86	1,12
1945	0,23	0,77	1,58	1,50	1,35	1,07	0,06	0,81	1,19
1934/37	2,13	0,17	3,17	-0,25	2,23	0,97	1,34	0,75	1,17
All Luxemburgian sets									
Number of data sets: 5									
Estimated mean	0,74	0,84	1,87	1,18	1,37	0,99	-0,07	0,79	1,19
Standard error of the estimate	0,35	0,20	0,33	0,41	0,23	0,07	0,40	0,02	0,03
t-statistic	2,12	4,16	5,70	2,89	-1,64	0,18	-0,18	10,10	-7,58
P-value (two-sided for relaxation)	0,0504	0,0070	0,0023	0,0223	0,1773	0,8635	0,8632	0,0003	0,0008

Norway	Initializations				Relaxations			Selections	
	Iterations		Initial ρ		Iterations	"Ratio"	ρ	Iterations	"Ratio"
	Differences		Differences		Ratio	Ratio	Difference	Ratio	Ratio
Year	N - Q	Q - A	N - Q	Q - A	P/C	P/C	P - C	O/R	O/R
1993	1,64	1,40	2,59	13,25	0,59	2,27	7,66	0,93	1,16
1989	2,41	1,42	2,09	11,42	0,54	2,10	3,22	0,84	1,26
1985	1,78	0,31	3,67	12,83	0,49	2,92	9,00	0,87	1,15
1981	1,57	3,46	3,76	13,25	0,58	2,62	8,83	0,80	1,31
1977	2,19	1,46	6,51	9,17	0,68	2,06	8,11	0,81	1,31
1973	1,19	3,75	5,42	10,59	0,60	1,83	2,89	0,91	1,10
1969	1,08	2,77	4,59	10,58	0,51	2,57	1,83	0,90	1,13
1965	1,96	1,71	4,33	10,92	0,45	2,84	4,50	0,93	1,09
1961	0,96	1,84	5,09	10,09	0,50	2,23	1,95	0,90	1,12
1957	0,54	1,27	5,17	11,33	0,43	2,73	4,17	0,92	1,10
1953	1,13	1,73	5,08	10,67	0,39	2,96	3,00	0,93	1,17
1949	1,44	2,63	7,34	10,84	0,37	2,77	-1,78	0,94	1,13
All Norwegian sets									
Number of data sets: 12									
Estimated mean	1,49	1,98	4,63	11,24	0,51	2,49	4,45	0,89	1,17
Standard error of the estimate	0,16	0,29	0,43	0,37	0,03	0,11	0,96	0,01	0,02
t-statistic	9,50	6,94	10,73	30,68	18,29	-13,69	4,63	8,26	-7,40
P-value (two-sided for relaxation)	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0007	0,0000	0,0000

Algorithm tests

Sweden	Initializations				Relaxations			Selections	
	Iterations		Initial ρ		Iterations	"Ratio"	ρ	Iterations	"Ratio"
	Differences		Differences		Ratio	Ratio	Difference	Ratio	Ratio
Year	N - Q	Q - A	N - Q	Q - A	P/C	P/C	P - C	O/R	O/R
1994	-0,04	5,25	0,66	29,84	0,46	3,02	13,84	0,87	1,29
1991	1,52	3,67	2,25	24,34	0,45	3,42	13,67	0,87	1,21
1988	0,57	2,08	1,67	23,59	0,41	3,10	9,67	0,91	1,09
1985	0,98	2,27	1,34	15,25	0,36	3,62	9,28	0,99	1,11
1982	-0,14	2,29	0,50	20,34	0,35	3,58	8,83	0,83	1,40
1979	1,88	3,09	0,50	18,00	0,40	3,37	8,05	0,91	1,26
1976	0,07	3,67	1,34	19,25	0,38	3,52	7,39	0,94	1,18
1973	0,96	4,21	1,17	19,84	0,36	3,85	9,22	0,84	1,43
1970	2,19	2,00	2,09	18,50	0,41	3,52	9,28	0,87	1,35
1968	1,27	3,19	1,09	18,58	0,42	3,50	12,67	0,88	1,20
1964	0,15	3,82	1,00	15,50	0,43	3,51	9,22	0,85	1,31
1960	0,02	3,46	1,34	13,17	0,35	3,52	5,11	0,92	1,21
1958	1,63	2,54	1,75	9,92	0,30	3,46	2,44	0,93	1,10
1956	1,79	2,98	2,84	12,08	0,30	3,56	2,67	0,92	1,21
1952	1,40	2,55	1,42	14,83	0,27	4,34	3,67	0,96	1,05
1948	1,30	2,10	1,59	14,91	0,26	3,79	2,56	0,93	1,15
1944	-0,26	3,31	2,84	13,17	0,25	3,72	-0,83	0,90	1,27
1940	2,73	2,02	4,34	14,17	0,32	3,71	4,33	0,84	1,17
1936	1,69	1,96	1,92	13,50	0,40	3,27	7,89	0,94	1,12
1932	0,15	2,47	3,50	11,42	0,34	3,80	4,83	0,86	1,52
1928	1,08	3,88	2,92	14,00	0,34	3,66	5,11	0,89	1,35
1924	2,71	4,13	3,92	15,08	0,36	3,53	10,11	0,85	1,23
1921	1,53	2,21	4,01	9,25	0,36	3,44	3,22	0,90	1,31
1920	2,59	2,98	4,92	15,42	0,17	4,29	-7,00	1,02	1,25
1917	2,80	1,11	3,50	13,25	0,15	5,88	-3,45	0,96	1,10
1914 II	2,92	9,96	4,25	10,50	0,12	4,58	-9,44	0,94	1,13
1914 I	2,86	10,82	2,67	10,00	0,14	2,75	-12,00	0,93	1,12
1911	3,90	8,25	5,83	9,75	0,13	6,64	-5,33	0,99	1,07
All Swedish sets									
Number of data sets: 28									
Estimated mean	1,43	3,65	2,40	15,62	0,32	3,78	4,46	0,91	1,22
Standard error of the estimate	0,21	0,44	0,27	0,92	0,02	0,15	1,28	0,01	0,02
t-statistic	6,78	8,30	8,88	16,96	35,57	-18,44	3,49	9,89	-9,97
P-value (two-sided for relaxation)	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0017	0,0000	0,0000
(1921 - 1994)									
Number of data sets: 23									
Estimated mean	1,09	3,00	2,00	16,46	0,36	3,56	7,05	0,90	1,24
Standard error of the estimate	0,19	0,19	0,24	1,02	0,01	0,06	0,81	0,01	0,02
t-statistic	5,84	16,14	8,41	16,12	53,26	-46,24	8,72	11,82	-9,79
P-value (two-sided for relaxation)	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000

All countries	Initializations				Relaxations			Selections	
	Iterations		Initial ρ		Iterations	"Ratio"	ρ	Iterations	"Ratio"
	Differences		Differences		Ratio	Ratio	Difference	Ratio	Ratio
	N - Q	Q - A	N - Q	Q - A	P/C	P/C	P - C	O/R	O/R
All sets									
Number of data sets: 104									
Estimated mean	1,00	2,50	2,86	10,07	0,58	2,48	3,65	0,86	1,22
Standard error of the estimate	0,09	0,16	0,20	0,56	0,03	0,10	0,51	0,01	0,01
t-statistic	11,34	15,34	14,63	18,10	14,46	-14,52	7,12	19,55	-19,14
P-value (two-sided for relaxation)	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000

Appendix 4:

Various data

and

Results of matrix bias tests

Description of Various data tables

These tables consist of three parts:

The **first part** presents the data sets utilized in the matrix bias test and their size.

The **second part** presents the following data regarding the execution of the fair share algorithm:

- Number of iterations of Algorithm 16.1 needed for the total discrepancy to satisfy the predecided tolerance level.
- The final total discrepancy

The **third part** presents GAMS data regarding the controlled rounding of the internal entries (LF apportionment). The most interesting figure here is the number of iterations GAMS uses to find the optimal rounding.

Description of Matrix bias tables

These tables are almost similar to the Vector bias tables. They show the matrix bias data for the comparison of group V and W when there are c groups. $V < W \leq c$, where $c = 2, 3$, or 4.

Matrix bias data are shown for all combinations of the 5 matrix apportionment methods; LF (controlled rounding), SD, DM, MF, and HA, and the 4 division methods; Cluster, Quota, Size, and Number. The formulas for $\varepsilon(V,W)$ and T are presented in section 5.8.

Various data

Data sets utilized in the *main* matrix bias test		Execution data for the fair share algorithm with a tolerance level of 1,0E-4		GAMS data regarding the work to find the optimal controlled rounding of the internal entries (LF apportionment)		
Data set	Size	Iterations	Final discrepancy	Iterations	Work space allocated	Solution time
Luxembourg 1954	4 x 5	6	2,4E-5	20	0,05 Mb	0,030 sec.
Iceland 1959	8 x 5	6	9,5E-5	40	0,06 Mb	0,180 sec.
Iceland 1978	8 x 5	6	1,4E-5	51	0,06 Mb	0,070 sec.
Iceland 1995	8 x 6	5	3,0E-5	42	0,07 Mb	0,200 sec.
Austria 1994	9 x 5	4	8,5E-5	57	0,06 Mb	0,190 sec.
Finland 1945	15 x 8	7	7,3E-5	118	0,11 Mb	0,090 sec.
Finland 1970	15 x 7	6	3,5E-5	142	0,10 Mb	0,080 sec.
Finland 1995	14 x 10	7	1,4E-5	191	0,13 Mb	0,110 sec.
Denmark 1966	23 x 7	7	1,5E-5	191	0,14 Mb	0,390 sec.
Denmark 1979	17 x 11	6	8,0E-6	296	0,16 Mb	0,150 sec.
Denmark 1994	17 x 9	5	1,8E-5	208	0,14 Mb	0,150 sec.
Norway 1949	29 x 9	10	9,7E-5	194	0,22 Mb	0,630 sec.
Norway 1969	20 x 8	16	7,9E-5	170	0,14 Mb	0,140 sec.
Norway 1993	19 x 8	6	1,3E-5	193	0,14 Mb	0,470 sec.
Germany 1919	36 x 13	11	5,5E-5	299	0,40 Mb	1,180 sec.
W. Germany 1949	9 x 10	10	9,1E-5	57	0,09 Mb	0,080 sec.
W. Germany 1972	10 x 3	4	1,6E-5	33	0,06 Mb	0,040 sec.
Germany 1994	16 x 5	7	1,6E-5	107	0,09 Mb	0,080 sec.
Sweden 1911	56 x 3	6	3,0E-5	201	0,17 Mb	0,460 sec.
Sweden 1928	28 x 6	6	1,9E-5	190	0,15 Mb	0,120 sec.
Sweden 1948	28 x 5	6	1,0E-5	176	0,13 Mb	0,370 sec.
Sweden 1970	28 x 6	5	1,2E-5	214	0,15 Mb	0,110 sec.
Sweden 1994	24 x 8	4	7,9E-5	295	0,17 Mb	0,470 sec.
Sum		156	9,3E-4	3485		
Average		6,78	4,0E-5	151,52		

Matrix bias 2 groups

ε (1,2)

Bias between group 1 and 2 for $c = 2$ based on 23 data sets

	LF	SD	DM	MF	HA
		Cluster division			
ε	-0,48 %	-5,81 %	-2,46 %	-1,97 %	1,72 %
S	0,66 %	2,92 %	0,87 %	0,81 %	1,90 %
T	-0,72	-1,99	-2,84 **	-2,43 *	0,90
P	47,7 %	5,9 %	1,0 %	2,4 %	37,6 %
		Quota division			
ε	-0,73 %	-4,36 %	-2,56 %	-1,62 %	0,01 %
S	0,54 %	1,62 %	0,88 %	0,82 %	1,28 %
T	-1,34	-2,69 *	-2,91 **	-1,97	0,01
P	19,4 %	1,4 %	0,8 %	6,1 %	99,2 %
		Size division			
ε	-0,21 %	3,03 %	-0,59 %	-3,10 %	-4,95 %
S	0,48 %	1,97 %	0,72 %	1,21 %	1,81 %
T	-0,44	1,54	-0,83	-2,57 *	-2,74 *
P	66,4 %	13,8 %	41,6 %	1,7 %	1,2 %
		Number division			
ε	8,00 %	-26,13 %	5,86 %	11,69 %	28,23 %
S	5,48 %	9,23 %	5,65 %	5,71 %	5,25 %
T	1,46	-2,83 **	1,04	2,05	5,38 **
P	15,9 %	1,0 %	31,1 %	5,3 %	0,0 %

ε Estimated bias percentage (mean)
S Standard error of the estimate
T t-statistic
P P-value (probability); two-sided for all methods

** Significant at the 1%-level
* Significant at the 5%-level

Matrix bias 3 groups

ε (1,2)

Bias between group 1 and 2 for c = 3 based on 23 data sets

	LF	SD	DM	MF	HA
		Cluster division			
ε	-0,79 %	9,04 %	0,43 %	-4,53 %	-8,77 %
S	0,64 %	2,22 %	0,90 %	1,51 %	2,17 %
T	-1,24	4,07 **	0,48	-3,00 **	-4,03 **
P	22,7 %	0,1 %	63,4 %	0,7 %	0,1 %
		Quota division			
ε	-0,55 %	6,11 %	-1,39 %	-2,80 %	-7,14 %
S	0,63 %	1,70 %	0,84 %	1,12 %	1,42 %
T	-0,87	3,60 **	-1,67	-2,50 *	-5,03 **
P	39,5 %	0,2 %	11,0 %	2,0 %	0,0 %
		Size division			
ε	-0,79 %	9,09 %	0,86 %	-4,33 %	-7,97 %
S	0,73 %	2,54 %	1,01 %	1,61 %	2,13 %
T	-1,08	3,58 **	0,86	-2,68 *	-3,74 **
P	29,3 %	0,2 %	40,2 %	1,4 %	0,1 %
		Number division			
ε	-3,50 %	-29,46 %	-10,17 %	-3,64 %	4,31 %
S	4,12 %	11,29 %	4,05 %	5,19 %	6,12 %
T	-0,85	-2,61 *	-2,51 *	-0,70	0,70
P	40,4 %	1,6 %	2,0 %	49,1 %	48,8 %

ε (1,3)

Bias between group 1 and 3 for c = 3 based on 23 data sets

	LF	SD	DM	MF	HA
		Cluster division			
ε	-1,66 %	-10,80 %	-5,43 %	-4,54 %	1,94 %
S	0,90 %	4,36 %	1,46 %	1,58 %	2,75 %
T	-1,85	-2,48 *	-3,71 **	-2,87 **	0,70
P	7,8 %	2,1 %	0,1 %	0,9 %	48,9 %
		Quota division			
ε	-0,68 %	-9,23 %	-3,51 %	-2,02 %	2,50 %
S	0,66 %	2,24 %	1,03 %	0,73 %	1,45 %
T	-1,03	-4,12 **	-3,39 **	-2,76 *	1,73
P	31,6 %	0,0 %	0,3 %	1,1 %	9,8 %
		Size division			
ε	-0,80 %	0,52 %	-3,40 %	-5,20 %	-4,56 %
S	0,88 %	2,87 %	1,22 %	1,62 %	2,25 %
T	-0,90	0,18	-2,77 *	-3,20 **	-2,03
P	37,6 %	85,8 %	1,1 %	0,4 %	5,5 %
		Number division			
ε	29,04 %	-5,55 %	30,53 %	38,77 %	56,44 %
S	7,85 %	12,52 %	9,62 %	8,92 %	8,48 %
T	3,70 **	-0,44	3,17 **	4,35 **	6,65 **
P	0,1 %	66,2 %	0,4 %	0,0 %	0,0 %

Matrix bias 3 groups

ε (2,3)

Bias between group 2 and 3 for $c = 3$ based on 23 data sets

	LF	SD	DM	MF	HA
		Cluster division			
ε	-0,89 %	-22,78 %	-6,07 %	-0,11 %	9,69 %
S	0,82 %	4,72 %	1,74 %	1,07 %	2,24 %
T	-1,09	-4,83 **	-3,49 **	-0,10	4,33 **
P	28,6 %	0,0 %	0,2 %	92,0 %	0,0 %
		Quota division			
ε	-0,24 %	-17,22 %	-2,22 %	0,60 %	8,81 %
S	1,01 %	3,21 %	1,28 %	0,88 %	1,41 %
T	-0,23	-5,36 **	-1,73	0,68	6,24 **
P	81,7 %	0,0 %	9,8 %	50,5 %	0,0 %
		Size division			
ε	-0,05 %	-10,81 %	-4,42 %	-0,97 %	2,84 %
S	0,76 %	3,60 %	1,28 %	1,17 %	2,00 %
T	-0,06	-3,00 **	-3,47 **	-0,82	1,42
P	95,3 %	0,7 %	0,2 %	41,9 %	16,9 %
		Number division			
ε	23,84 %	6,14 %	29,94 %	26,17 %	-2,17 %
S	10,54 %	12,95 %	10,59 %	14,63 %	49,27 %
T	2,26 *	0,47	2,83 **	1,79	-0,04
P	3,4 %	64,0 %	1,0 %	8,7 %	96,5 %

ε Estimated bias percentage (mean)
S Standard error of the estimate
T t-statistic
P P-value (probability); two-sided for all methods

** Significant at the 1%-level
* Significant at the 5%-level

Matrix bias 4 groups

ε (1,2)

Bias between group 1 and 2 for c = 4 based on 23 data sets (21 for Size division)

	LF	SD	DM	MF	HA
		Cluster division			
ε	-0,97 %	7,73 %	-0,08 %	-5,13 %	-6,84 %
S	0,73 %	1,98 %	0,75 %	1,53 %	1,81 %
T	-1,32	3,90 **	-0,10	-3,35 **	-3,78 **
P	19,9 %	0,1 %	92,0 %	0,3 %	0,1 %
		Quota division			
ε	0,07 %	4,36 %	-0,69 %	-2,51 %	-5,26 %
S	0,84 %	1,79 %	0,89 %	1,30 %	1,59 %
T	0,09	2,44 *	-0,77	-1,93	-3,32 **
P	93,0 %	2,3 %	44,7 %	6,7 %	0,3 %
		Size division			
ε	-0,76 %	3,90 %	-1,20 %	-3,29 %	-4,35 %
S	0,99 %	1,30 %	0,93 %	1,57 %	1,37 %
T	-0,76	2,99 **	-1,29	-2,10 *	-3,16 **
P	45,5 %	0,7 %	21,1 %	4,9 %	0,5 %
		Number division			
ε	-2,52 %	-9,46 %	-5,23 %	-4,36 %	-1,67 %
S	2,17 %	4,67 %	2,50 %	2,14 %	2,87 %
T	-1,16	-2,03	-2,09 *	-2,04	-0,58
P	25,8 %	5,5 %	4,8 %	5,4 %	56,7 %

ε (1,3)

Bias between group 1 and 3 for c = 4 based on 23 data sets (21 for Size division)

	LF	SD	DM	MF	HA
		Cluster division			
ε	0,85 %	11,21 %	0,89 %	-4,30 %	-11,35 %
S	0,87 %	2,52 %	1,21 %	1,61 %	2,38 %
T	0,98	4,46 **	0,73	-2,67 *	-4,76 **
P	33,8 %	0,0 %	47,0 %	1,4 %	0,0 %
		Quota division			
ε	0,99 %	9,98 %	0,31 %	-2,10 %	-9,68 %
S	0,69 %	2,34 %	1,06 %	1,37 %	1,85 %
T	1,43	4,27 **	0,29	-1,54	-5,22 **
P	16,6 %	0,0 %	77,6 %	13,9 %	0,0 %
		Size division			
ε	-0,14 %	8,46 %	1,61 %	-3,95 %	-10,16 %
S	0,95 %	2,07 %	1,10 %	1,73 %	2,25 %
T	-0,14	4,09 **	1,47	-2,29 *	-4,51 **
P	88,7 %	0,1 %	15,7 %	3,3 %	0,0 %
		Number division			
ε	-0,66 %	-31,52 %	-5,34 %	-0,08 %	15,29 %
S	6,05 %	9,15 %	6,45 %	6,40 %	5,74 %
T	-0,11	-3,45 **	-0,83	-0,01	2,66 *
P	91,5 %	0,2 %	41,7 %	99,0 %	1,4 %

Matrix bias 4 groups

ε (1,4)

Bias between group 1 and 4 for $c = 4$ based on 23 data sets (21 for Size division)

	LF	SD	DM	MF	HA
		Cluster division			
ε	-3,04 %	-21,03 %	-8,31 %	-4,24 %	9,56 %
S	1,39 %	6,27 %	1,85 %	2,31 %	3,90 %
T	-2,19 *	-3,35 **	-4,48 **	-1,83	2,45 *
P	4,0 %	0,3 %	0,0 %	8,1 %	2,3 %
		Quota division			
ε	-2,25 %	-14,72 %	-6,07 %	-3,43 %	4,77 %
S	1,06 %	2,90 %	1,34 %	1,32 %	1,69 %
T	-2,12 *	-5,08 **	-4,54 **	-2,59 *	2,81 *
P	4,5 %	0,0 %	0,0 %	1,7 %	1,0 %
		Size division			
ε	-1,42 %	-3,19 %	-4,46 %	-5,63 %	-2,98 %
S	1,12 %	4,21 %	1,77 %	1,84 %	3,07 %
T	-1,27	-0,76	-2,53 *	-3,06 **	-0,97
P	22,0 %	45,7 %	2,0 %	0,6 %	34,4 %
		Number division			
ε	36,11 %	2,21 %	42,30 %	48,91 %	64,01 %
S	9,22 %	14,76 %	11,36 %	10,33 %	8,68 %
T	3,92 **	0,15	3,72 **	4,74 **	7,38 **
P	0,1 %	88,2 %	0,1 %	0,0 %	0,0 %

ε (2,3)

Bias between group 2 and 3 for $c = 4$ based on 23 data sets (21 for Size division)

	LF	SD	DM	MF	HA
		Cluster division			
ε	1,71 %	3,50 %	0,90 %	0,65 %	-4,33 %
S	1,02 %	2,35 %	1,23 %	1,21 %	1,78 %
T	1,67	1,49	0,73	0,54	-2,44 *
P	10,9 %	15,0 %	47,5 %	59,3 %	2,3 %
		Quota division			
ε	0,79 %	5,59 %	0,80 %	0,25 %	-4,48 %
S	0,96 %	2,28 %	1,41 %	1,21 %	1,91 %
T	0,82	2,45 *	0,57	0,20	-2,35 *
P	42,1 %	2,3 %	57,3 %	84,1 %	2,8 %
		Size division			
ε	0,52 %	4,50 %	2,64 %	-0,88 %	-5,70 %
S	0,94 %	2,31 %	1,31 %	1,78 %	2,05 %
T	0,55	1,95	2,01	-0,50	-2,79 *
P	59,0 %	6,5 %	5,8 %	62,4 %	1,1 %
		Number division			
ε	0,07 %	-25,80 %	-2,74 %	2,41 %	16,37 %
S	6,76 %	9,55 %	6,92 %	6,90 %	6,44 %
T	0,01	-2,70 *	-0,40	0,35	2,54 *
P	99,2 %	1,3 %	69,6 %	73,0 %	1,9 %

Matrix bias 4 groups

ε (2,4)

Bias between group 2 and 4 for c = 4 based on 23 data sets (21 for Size division)

	LF	SD	DM	MF	HA
		Cluster division			
ε	-2,10 %	-33,14 %	-8,35 %	0,88 %	15,16 %
S	1,35 %	7,95 %	1,98 %	1,60 %	3,69 %
T	-1,56	-4,17 **	-4,21 **	0,55	4,11 **
P	13,4 %	0,0 %	0,0 %	58,9 %	0,0 %
		Quota division			
ε	-2,39 %	-20,76 %	-5,46 %	-0,99 %	9,29 %
S	1,02 %	3,54 %	1,39 %	0,93 %	1,65 %
T	-2,35 *	-5,86 **	-3,93 **	-1,06	5,63 **
P	2,8 %	0,0 %	0,1 %	30,1 %	0,0 %
		Size division			
ε	-0,75 %	-7,35 %	-3,27 %	-2,49 %	1,19 %
S	1,06 %	4,00 %	1,59 %	1,81 %	2,94 %
T	-0,71	-1,84	-2,06	-1,37	0,40
P	48,5 %	8,1 %	5,3 %	18,5 %	69,1 %
		Number division			
ε	36,58 %	3,94 %	43,47 %	49,67 %	60,16 %
S	9,40 %	15,84 %	11,31 %	10,29 %	10,35 %
T	3,89 **	0,25	3,84 **	4,83 **	5,81 **
P	0,1 %	80,6 %	0,1 %	0,0 %	0,0 %

ε (3,4)

Bias between group 3 and 4 for c = 4 based on 23 data sets (21 for Size division)

	LF	SD	DM	MF	HA
		Cluster division			
ε	-4,27 %	-38,71 %	-9,77 %	-0,28 %	18,63 %
S	2,13 %	7,86 %	2,63 %	2,45 %	3,36 %
T	-2,01	-4,93 **	-3,71 **	-0,11	5,54 **
P	5,7 %	0,0 %	0,1 %	91,0 %	0,0 %
		Quota division			
ε	-3,38 %	-30,05 %	-6,71 %	-1,49 %	12,81 %
S	1,27 %	5,53 %	1,93 %	1,29 %	1,75 %
T	-2,67 *	-5,43 **	-3,48 **	-1,15	7,34 **
P	1,4 %	0,0 %	0,2 %	26,3 %	0,0 %
		Size division			
ε	-1,42 %	-13,20 %	-6,52 %	-1,87 %	6,54 %
S	1,35 %	4,32 %	2,46 %	1,81 %	2,10 %
T	-1,05	-3,06 **	-2,65 *	-1,03	3,11 **
P	30,5 %	0,6 %	1,6 %	31,4 %	0,5 %
		Number division			
ε	-11,05 %	-23,12 %	-5,06 %	-4,89 %	6,13 %
S	46,52 %	46,32 %	47,30 %	48,34 %	48,81 %
T	-0,24	-0,50	-0,11	-0,10	0,13
P	81,4 %	62,3 %	91,6 %	92,0 %	90,1 %

Various data (Iceland)

Data sets utilized in the *Icelandic* matrix bias test		Execution data for the fair share algorithm with a tolerance level of 1,0E-4		GAMS data regarding the work to find the optimal controlled rounding of the internal entries (LF apportionment)		
Data set	Size	Iterations	Final discrepancy	Iterations	Work space allocated	Solution time
Iceland 1959	8 x 5	6	9,5E-5	40	0,06 Mb	0,180 sec.
Iceland 1963	8 x 4	6	1,3E-5	31	0,06 Mb	0,050 sec.
Iceland 1967	8 x 6	6	1,9E-5	46	0,07 Mb	0,030 sec.
Iceland 1971	8 x 6	5	6,5E-5	55	0,07 Mb	0,070 sec.
Iceland 1974	8 x 5	5	3,1E-5	51	0,06 Mb	0,040 sec.
Iceland 1978	8 x 5	6	1,4E-5	51	0,06 Mb	0,070 sec.
Iceland 1979	8 x 5	8	3,6E-5	38	0,06 Mb	0,030 sec.
Iceland 1983	8 x 6	6	2,3E-5	38	0,07 Mb	0,060 sec.
Iceland 1987	8 x 9	8	3,3E-5	69	0,08 Mb	0,080 sec.
Iceland 1991	8 x 7	5	8,6E-5	44	0,07 Mb	0,040 sec.
Iceland 1995	8 x 6	5	3,0E-5	42	0,07 Mb	0,200 sec.
Sum		66	4,5E-4	505		
Average		6,00	4,0E-5	45,91		

Matrix bias 2 groups (Iceland)

ε (1,2)

Bias between group 1 and 2 for $c = 2$ based on 11 Icelandic data sets

	LF	SD	DM	MF	HA
		Cluster division			
ε	-1,64 %	2,41 %	-2,36 %	-3,51 %	-5,50 %
S	0,96 %	1,91 %	1,74 %	1,79 %	1,48 %
T	-1,70	1,26	-1,36	-1,96	-3,71 **
P	12,0 %	23,5 %	20,4 %	7,9 %	0,4 %
		Quota division			
ε	-0,44 %	-2,95 %	-4,85 %	-4,96 %	-2,95 %
S	0,67 %	1,97 %	1,28 %	1,85 %	2,02 %
T	-0,65	-1,50	-3,78 **	-2,69 *	-1,46
P	53,1 %	16,5 %	0,4 %	2,3 %	17,4 %
		Size division			
ε	-1,80 %	4,82 %	-0,25 %	-3,69 %	-5,14 %
S	1,10 %	1,68 %	1,50 %	2,05 %	2,24 %
T	-1,64	2,87 *	-0,17	-1,80	-2,29 *
P	13,3 %	1,7 %	86,9 %	10,3 %	4,5 %
		Number division			
ε	4,10 %	-29,25 %	-1,67 %	7,58 %	24,97 %
S	6,57 %	10,46 %	7,80 %	7,84 %	5,60 %
T	0,62	-2,80 *	-0,21	0,97	4,46 **
P	54,6 %	1,9 %	83,4 %	35,6 %	0,1 %

ε Estimated bias percentage (mean)
S Standard error of the estimate
T t-statistic
P P-value (probability); two-sided for all methods

** Significant at the 1%-level
* Significant at the 5%-level

Matrix bias 3 groups (Iceland)

ε (1,2)

Bias between group 1 and 2 for c = 3 based on 11 Icelandic data sets

	LF	SD	DM	MF	HA
		Cluster division			
ε	-1,62 %	13,07 %	2,28 %	-6,57 %	-5,76 %
S	1,68 %	3,21 %	2,02 %	3,42 %	2,37 %
T	-0,96	4,07 **	1,13	-1,92	-2,43 *
P	35,9 %	0,2 %	28,4 %	8,3 %	3,5 %
		Quota division			
ε	-1,32 %	9,33 %	-1,49 %	-4,78 %	-6,78 %
S	1,37 %	3,15 %	1,62 %	2,66 %	2,99 %
T	-0,96	2,96 *	-0,92	-1,80	-2,27 *
P	35,8 %	1,4 %	38,0 %	10,3 %	4,7 %
		Size division			
ε	0,08 %	12,90 %	4,69 %	-5,32 %	-4,47 %
S	1,88 %	4,99 %	2,94 %	3,78 %	2,53 %
T	0,04	2,58 *	1,60	-1,41	-1,77
P	96,5 %	2,7 %	14,2 %	18,9 %	10,7 %
		Number division			
ε	-8,49 %	-26,07 %	-18,69 %	-14,62 %	-9,55 %
S	3,76 %	6,57 %	5,07 %	3,44 %	2,93 %
T	-2,26 *	-3,97 **	-3,69 **	-4,25 **	-3,26 **
P	4,8 %	0,3 %	0,4 %	0,2 %	0,9 %

ε (1,3)

Bias between group 1 and 3 for c = 3 based on 11 Icelandic data sets

	LF	SD	DM	MF	HA
		Cluster division			
ε	-1,87 %	-3,83 %	-2,63 %	-7,49 %	-4,26 %
S	1,55 %	2,91 %	1,48 %	2,60 %	1,68 %
T	-1,20	-1,32	-1,77	-2,88 *	-2,54 *
P	25,6 %	21,7 %	10,7 %	1,6 %	3,0 %
		Quota division			
ε	-0,93 %	-5,51 %	-3,57 %	-3,31 %	-3,03 %
S	1,11 %	1,88 %	1,29 %	1,30 %	2,01 %
T	-0,84	-2,93 *	-2,78 *	-2,54 *	-1,51
P	42,2 %	1,5 %	1,9 %	2,9 %	16,1 %
		Size division			
ε	-1,27 %	5,47 %	-1,73 %	-8,07 %	-7,24 %
S	1,37 %	2,42 %	1,73 %	2,90 %	2,42 %
T	-0,93	2,26 *	-1,00	-2,78 *	-2,99 *
P	37,6 %	4,7 %	34,0 %	1,9 %	1,4 %
		Number division			
ε	15,24 %	-4,90 %	50,20 %	61,93 %	79,00 %
S	17,18 %	21,46 %	17,68 %	13,57 %	9,03 %
T	0,89	-0,23	2,84 *	4,56 **	8,75 **
P	39,6 %	82,4 %	1,8 %	0,1 %	0,0 %

Matrix bias 3 groups (Iceland)

ε (2,3)

Bias between group 2 and 3 for $c = 3$ based on 11 Icelandic data sets

	LF	SD	DM	MF	HA
		Cluster division			
ε	-0,40 %	-20,81 %	-5,25 %	-1,36 %	1,11 %
S	1,50 %	5,10 %	1,39 %	2,14 %	1,93 %
T	-0,27	-4,08 **	-3,77 **	-0,64	0,58
P	79,5 %	0,2 %	0,4 %	54,0 %	57,7 %
		Quota division			
ε	0,17 %	-17,69 %	-2,35 %	0,91 %	3,10 %
S	1,89 %	4,20 %	2,19 %	2,14 %	1,87 %
T	0,09	-4,22 **	-1,07	0,42	1,66
P	93,0 %	0,2 %	30,9 %	68,1 %	12,8 %
		Size division			
ε	-1,58 %	-12,32 %	-7,40 %	-3,34 %	-3,04 %
S	1,55 %	7,14 %	2,39 %	2,78 %	2,60 %
T	-1,02	-1,72	-3,09 *	-1,20	-1,17
P	33,3 %	11,5 %	1,1 %	25,7 %	26,9 %
		Number division			
ε	16,59 %	9,60 %	51,26 %	64,24 %	80,59 %
S	20,35 %	20,71 %	17,44 %	12,71 %	8,61 %
T	0,82	0,46	2,94 *	5,06 **	9,37 **
P	43,4 %	65,3 %	1,5 %	0,0 %	0,0 %

ε Estimated bias percentage (mean)
S Standard error of the estimate
T t-statistic
P P-value (probability); two-sided for all methods

** Significant at the 1%-level
* Significant at the 5%-level

Matrix bias 4 groups (Iceland)

ε (1,2)

Bias between group 1 and 2 for $c = 4$ based on 11 Icelandic data sets (6 for Size division)

	LF	SD	DM	MF	HA
		Cluster division			
ε	0,81 %	8,91 %	3,32 %	-5,62 %	-4,12 %
S	2,03 %	2,81 %	2,49 %	3,79 %	1,97 %
T	0,40	3,17 *	1,33	-1,48	-2,09
P	69,7 %	1,0 %	21,3 %	16,9 %	6,3 %
		Quota division			
ε	-3,82 %	8,16 %	0,45 %	-3,82 %	-5,38 %
S	1,86 %	2,60 %	2,78 %	2,80 %	2,82 %
T	-2,05	3,14 *	0,16	-1,36	-1,91
P	6,7 %	1,0 %	87,4 %	20,3 %	8,6 %
		Size division			
ε	3,12 %	4,79 %	-0,87 %	-12,19 %	-5,67 %
S	3,65 %	4,43 %	3,84 %	3,55 %	1,92 %
T	0,85	1,08	-0,23	-3,43 *	-2,95 *
P	43,2 %	33,0 %	82,9 %	1,9 %	3,2 %
		Number division			
ε	-2,62 %	-1,65 %	-3,04 %	-5,01 %	-11,44 %
S	3,93 %	5,67 %	5,19 %	3,27 %	2,02 %
T	-0,67	-0,29	-0,59	-1,53	-5,68 **
P	52,0 %	77,7 %	57,1 %	15,7 %	0,0 %

ε (1,3)

Bias between group 1 and 3 for $c = 4$ based on 11 Icelandic data sets (6 for Size division)

	LF	SD	DM	MF	HA
		Cluster division			
ε	0,21 %	17,37 %	1,64 %	-7,67 %	-12,95 %
S	1,80 %	4,60 %	2,54 %	3,63 %	3,39 %
T	0,12	3,78 **	0,65	-2,11	-3,82 **
P	91,0 %	0,4 %	53,3 %	6,1 %	0,3 %
		Quota division			
ε	0,54 %	17,77 %	2,28 %	-5,55 %	-11,20 %
S	2,33 %	3,98 %	2,42 %	3,67 %	2,99 %
T	0,23	4,47 **	0,94	-1,51	-3,74 **
P	82,2 %	0,1 %	36,8 %	16,1 %	0,4 %
		Size division			
ε	-3,80 %	3,59 %	0,08 %	-15,18 %	-8,18 %
S	2,32 %	2,44 %	2,63 %	4,00 %	4,03 %
T	-1,64	1,47	0,03	-3,79 **	-2,03
P	16,3 %	20,2 %	97,6 %	1,3 %	9,8 %
		Number division			
ε	-7,12 %	-41,15 %	-17,13 %	-9,77 %	7,94 %
S	7,36 %	10,41 %	8,17 %	8,87 %	7,29 %
T	-0,97	-3,95 **	-2,10	-1,10	1,09
P	35,6 %	0,3 %	6,2 %	29,7 %	30,1 %

Matrix bias 4 groups (Iceland)

ε (1,4)

Bias between group 1 and 4 for c = 4 based on 11 Icelandic data sets (6 for Size division)

	LF	SD	DM	MF	HA
		Cluster division			
ε	-5,32 %	-18,57 %	-10,12 %	-10,11 %	4,53 %
S	1,80 %	4,20 %	1,94 %	2,15 %	3,91 %
T	-2,96 *	-4,42 **	-5,21 **	-4,71 **	1,16
P	1,4 %	0,1 %	0,0 %	0,1 %	27,3 %
		Quota division			
ε	-5,12 %	-14,81 %	-8,64 %	-6,93 %	-0,23 %
S	1,64 %	2,93 %	1,99 %	1,56 %	2,73 %
T	-3,11 *	-5,06 **	-4,35 **	-4,45 **	-0,08
P	1,1 %	0,0 %	0,1 %	0,1 %	93,5 %
		Size division			
ε	-2,40 %	2,52 %	-6,23 %	-13,62 %	-9,34 %
S	2,20 %	3,88 %	2,11 %	3,65 %	4,10 %
T	-1,09	0,65	-2,95 *	-3,74 *	-2,28
P	32,4 %	54,6 %	3,2 %	1,3 %	7,2 %
		Number division			
ε	51,18 %	12,98 %	64,83 %	74,97 %	85,54 %
S	15,86 %	22,10 %	13,90 %	11,17 %	10,07 %
T	3,23 **	0,59	4,67 **	6,71 **	8,49 **
P	0,9 %	57,0 %	0,1 %	0,0 %	0,0 %

ε (2,3)

Bias between group 2 and 3 for c = 4 based on 11 Icelandic data sets (6 for Size division)

	LF	SD	DM	MF	HA
		Cluster division			
ε	-0,90 %	9,33 %	-2,06 %	-2,29 %	-8,59 %
S	2,22 %	4,15 %	2,69 %	2,37 %	3,05 %
T	-0,40	2,25 *	-0,77	-0,97	-2,81 *
P	69,4 %	4,8 %	46,1 %	35,6 %	1,8 %
		Quota division			
ε	4,06 %	10,22 %	1,36 %	-1,71 %	-6,04 %
S	2,13 %	3,89 %	2,76 %	2,42 %	3,35 %
T	1,90	2,62 *	0,49	-0,71	-1,80
P	8,6 %	2,5 %	63,4 %	49,5 %	10,2 %
		Size division			
ε	-7,49 %	-2,04 %	0,48 %	-2,81 %	-2,39 %
S	2,04 %	3,98 %	3,25 %	2,71 %	3,40 %
T	-3,68 *	-0,51	0,15	-1,04	-0,70
P	1,4 %	63,0 %	88,8 %	34,6 %	51,5 %
		Number division			
ε	-7,33 %	-44,96 %	-18,91 %	-7,60 %	16,67 %
S	9,35 %	12,71 %	11,10 %	10,91 %	7,33 %
T	-0,78	-3,54 **	-1,70	-0,70	2,27 *
P	45,1 %	0,5 %	11,9 %	50,2 %	4,6 %

Matrix bias 4 groups (Iceland)

ε (2,4)

Bias between group 2 and 4 for c = 4 based on 11 Icelandic data sets (6 for Size division)

	LF	SD	DM	MF	HA
		Cluster division			
ε	-6,40 %	-31,88 %	-14,43 %	-5,21 %	8,19 %
S	1,76 %	7,47 %	2,80 %	3,19 %	3,52 %
T	-3,63 **	-4,27 **	-5,16 **	-1,63	2,32 *
P	0,5 %	0,2 %	0,0 %	13,3 %	4,3 %
		Quota division			
ε	-1,48 %	-25,70 %	-9,62 %	-3,52 %	4,76 %
S	1,91 %	3,84 %	2,23 %	2,26 %	1,66 %
T	-0,78	-6,69 **	-4,31 **	-1,56	2,86 *
P	45,6 %	0,0 %	0,2 %	15,0 %	1,7 %
		Size division			
ε	-6,09 %	-2,63 %	-5,71 %	-1,53 %	-3,49 %
S	2,46 %	2,12 %	2,24 %	3,24 %	3,45 %
T	-2,47	-1,24	-2,54	-0,47	-1,01
P	5,6 %	27,0 %	5,2 %	65,6 %	35,9 %
		Number division			
ε	49,15 %	6,83 %	63,27 %	74,53 %	86,54 %
S	16,57 %	24,17 %	14,32 %	11,39 %	9,26 %
T	2,97 *	0,28	4,42 **	6,54 **	9,34 **
P	1,4 %	78,3 %	0,1 %	0,0 %	0,0 %

ε (3,4)

Bias between group 3 and 4 for c = 4 based on 11 Icelandic data sets (6 for Size division)

	LF	SD	DM	MF	HA
		Cluster division			
ε	-5,97 %	-49,27 %	-12,92 %	-3,12 %	14,93 %
S	3,01 %	11,44 %	4,13 %	3,21 %	3,81 %
T	-1,98	-4,31 **	-3,13 *	-0,97	3,91 **
P	7,5 %	0,2 %	1,1 %	35,4 %	0,3 %
		Quota division			
ε	-6,44 %	-43,35 %	-11,70 %	-2,16 %	9,42 %
S	3,50 %	8,63 %	2,86 %	2,63 %	2,81 %
T	-1,84	-5,02 **	-4,10 **	-0,82	3,35 **
P	9,6 %	0,1 %	0,2 %	43,1 %	0,7 %
		Size division			
ε	1,30 %	-1,15 %	-6,53 %	1,27 %	-1,10 %
S	1,34 %	3,35 %	2,41 %	1,40 %	1,06 %
T	0,97	-0,34	-2,71 *	0,91	-1,04
P	37,7 %	74,6 %	4,2 %	40,4 %	34,6 %
		Number division			
ε	52,98 %	41,85 %	70,50 %	77,93 %	80,14 %
S	15,83 %	14,82 %	12,12 %	10,33 %	14,88 %
T	3,35 **	2,82 *	5,81 **	7,55 **	5,39 **
P	0,7 %	1,8 %	0,0 %	0,0 %	0,0 %