Decompositions of Travelling Salesman Problems

by

Øyvind Halskau

Doctoral Thesis

Hamilton's Icosian Game

Preface

This work is the author's doctor thesis for the dr. oecon. title.

Even if the work has been performed without formal supervision, it would not have been possible to complete the thesis without the support and inspiration from many people. Especially I would like to mention professor Kurt Jörnsten, NHH and associate professor Kjetil Haugen, HSM. Without their help, suggestions, and constructive discussions I would probably have given up somewhere along the path

I would also like to thank my employer, Høgskolen i Molde (HSM) for giving me time and sometimes money, not to mention access to a very good library and an equivalent computer support department. Without such facilities, the work would not have been possible.

Finally, I would like to thank my wife Ragnhild, not for her patience, but for her silent "whip", which gave me a freedom to work as much and as often as I wanted, without any bad conscience for my absence.

Molde - Norway

December 1999

Øyvind Halskau

Contents

Summary

This thesis deals with the Travelling Salesman Problem.

In chapter 1 some historical comments concerning the problem are offered and short overviews of different applications and variations of the problem are given. Further, some classical naïve heuristics for TSP are discussed together with the problem of finding lower bounds for the problem. In the last section some polynomial classes for TSP are overviewed and an alternative proof for the polynomial class of symmetric product matrices is given.

In chapter 2 the saving heuristic of Clarke and Wright is revisited. It is shown that the cost of any Hamiltonian cycle measured in the original cost matrix plus the cost of the same cycle measured in one of the saving matrices is a constant. This fact can be utilised in several ways. First of all the saving matrices can be used - in principle - as input to any to any heuristic for TSP, often yielding better solutions than those obtained by the original cost matrix alone. Secondly, the saving matrices in combination with constant-TSP can be used to obtain good lower bounds for the TSP in question, only using 1-trees. Combinations of the obtained lower bounds together with solutions of simple IP models can improve on these bounds. Further, the saving matrices are used to make minor extensions of known polynomial classes for TSP and a new class of such matrices is offered. Finally, subsets of matrices are identified, which have the property that all matrices in a subset have the same cost for each Hamiltonian cycle.

In chapter 3 the cost matrix of TSP instances are decomposed with the help of the spectral theorem. It turns out that in the symmetric case, this decomposition can be performed with the help of a certain number of symmetric product matrices, that is, polynomial solvable TSPs. In this case a new way of calculating a lower bound for the TSP is offered. In the asymmetric case the decomposition is more difficult and involves complex eigenvalues and eigenvectors and the decomposition involves matrices that are not product matrices.

In chapter 4 a TSP instance is decomposed in different ways than in the previous chapter. Basically, any asymmetric cost matrix are decomposed into a constant-TSP matrix taking care of some of the asymmetric aspects of the original cost matrix, a symmetric matrix and a residual asymmetric. This kind of decomposition leads to the concept of Hamiltonian symmetric matrices, which are asymmetric matrices where every cycle and its reverse have the same cost as will be the case for symmetric matrices. Several criteria for identifying such matrices are given. Further, the concept of an optimal pair of Hamiltonian cycles is introduced. An optimal pair of cycles is the pair of cycles where the cost of the cycle plus the cost of the reversed cycle is as small as possible. To find this pair of cycles can be done by solving a symmetric TSP. The decompositions in this chapter are used in different ways. Partly they can be used as algorithms for finding optimal solutions to asymmetric instances of TSP, partly they can be used for finding upper and lower bounds. The performance of the procedure will in many cases depend on the cost structure of the

original cost matrix and how this structure is reflected particularly in the residual matrix.

Chapter 5 contains examples for some of the results in the previous chapters. The effects on upper and lower bounds using the different decomposition schemes are illustrated. Further, examples are given on how some naïve heuristics perform when saving matrices are used as input. The chapter also contains 10 examples of asymmetric TSPs taken from a public library and exposed to the methods described in chapter 4. At the end of the chapter some further research is suggested.

Chapter 1

1.1 Introduction

The Travelling Salesman Problem (TSP) is a well-studied combinatorial optimisation problem, which can briefly be described as the problem of finding the cheapest tour or cycle between a certain number of cities, where every city shall be visited only once, and the traveller shall return to the city he started from. It turns out that TSP belongs to a class of mathematical problems which is easy to explain to a layman, but in general is very difficult to solve.

Historically the TSP goes back a long time. The problem was indirectly mentioned already by Leonhard (Leonid) Euler (1707 - 1783) in 1759, and a bit later by his younger contemporary Alexandre-Theophile Vandermonde (1735 - 1796) in 1771, discussing the so-called knight's tour. A knight's tour is connected to the 64 squares on a chessboard. The knight's tour is then the problem of finding a sequence of legal moves - that is standard moves for a knight - starting from any square, visiting every square only once and returning to the first square. The answer is affirmative, but the tour is not a straightforward one. A solution can be found in Biggs et alt., 1977. In 1736 Euler had solved and generalised the famous problem for the bridges of Königsberg. He characterised graphs where it is possible to start anywhere and visit all edges only once and return to the starting point. Such characterisations are not known for TSP. There is a fundamental difference between the Königsberg problem and the latter one. The former can be said to be an explorer's problem, that is every place and road shall be visited [illegible] night's tour we have a traveller's problem, that is, we want to visit all [illegible] once. In both cases we want to return home.

The next person known to have been interested in TSP and related problems in a more general setting is probably reverend Thomas Penyngton Kirkman (1806 - 1895). Kirkman is an almost forgotten - and probably wrongly so - mathematician. Being active from his forties and almost to his death at nearly 90, he published works in many different fields of mathematics. His first published article appeared in 1846 dealing with combinatorial problems and his last article appeared in 1895 the same year he died. During the almost 50 years he published articles, he always seemed to return to questions dealing with combinatorial problems, but also gave contribution to the theory of equations and he was probably the person that understood Galois theory best among his British contemporaries. He solved the problem of Steiner triple system six years *before* Steiner proposed it and gave substantial contribution to the understanding of polyhedral and knots. He participated in the so-called Grand Prix de Mathématique arranged by the Académie des Science in Paris in the late fifties with a paper dealing with polyhedral. No one got the prize, but he was mentioned favourably together with Jordan and Mathieu. Nonetheless he is best remembered today for a

mere trifle, that is, for his so-called "schoolgirl problem" forthcoming in 1850 as a challenge to gentlefolk to discover the solution for themselves:

> *Fifteen young ladies in a school walk out three abreast for seven days in succession: it is required to arrange them daily, so that not two shall walk twice abreast.*

His article on what we today call Hamiltonian cycles, was published in 1855, and he explains that what he calls "closed polygons" can be employed as a means of representing polyhedral. He considered the following problem: Given the graph of a polyhedron, is it possible to find a circuit passing through each vertex one and only one time? He claimed to have asserted sufficient condition for graphs to contain a solution to TSP, but his claim was wrong. However, his major achievement in this field of mathematics was to describe a general class of graphs, which do not possess Hamiltonian cycles. He gave a general proof of the fact that if a polyhedron has an odd number of vertices, and each face has an even number of edges, no Hamiltonian cycle can exist in the graph. An example of such a graph, given by Kirkman, is drawn in fig .1.1.

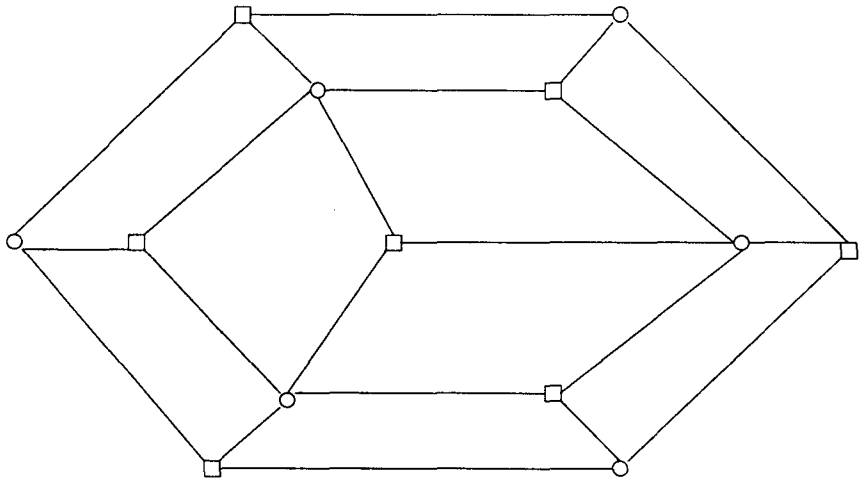


figure 1.1

In a more modern language the graph in fig. 1.1 is said to be *bipartite*, that is the vertices can be separated into two disjoint sets, such that any edge in the graph joins vertices in each of the two sets. In fig. 1.1 the two sets are indicated by circles and squares, respectively. Since a path in such a graph necessarily must visit vertices in the two sets alternatively, a cycle starting and ending in the same vertex must contain an even number of vertices. Hence, bipartite graphs having an odd number of vertices cannot contain Hamiltonian cycles.

It is interesting to note that a similar argument was used by Euler when he discussed the knight's tour on non- standard chessboards, that is boards with *nxn* squares. Defining the two sets of vertices as the black squares and the white squares, and observing that a knight must always move from a white square to a black one or vice versa, it follows that the knights tour is never possible on "chessboards" where n is an odd number

A very thorough and interesting article describing the life and work of the said reverend Kirkman can be found in Biggs, 1981.

Two years after Kirkman had published his paper on Hamiltionian cycles, the Irish mathematician William Rowan Hamilton (1805 - 1865) created a game the objective of which basically was to find cycles in specific graphs. Hamilton was generally recognised as one of the leading mathematicians of his time, giving major contributions to non-commutative algebra, optical geometry and dynamics. The game invented by Hamilton was called *The Icosian Game* (ico is Greek for twenty). He sold the game to a wholesale dealer in games and puzzles for £25. The basic idea was to use a specified graph - see figure 1.2 - to create different sub-graphs, and amongst other things, cycles containing all the vertices.

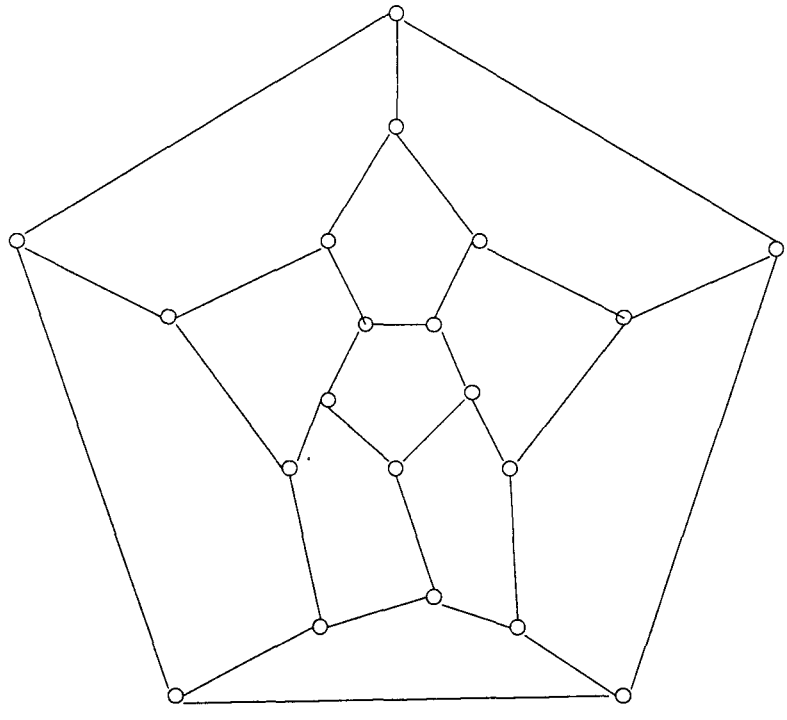


figure 1.2

Another version of this (planar) game involved a solid dodecahedron, known as " The traveller's Dodecahedron" or "A Voyage Round the World". Each vertex was given a name after some major city in the world, starting with Brussels and ending with Zanzibar. All the cities were marked by pegs, and a thread should be looped around the pegs in order to indicate the itinerary. A complete cycle - a Hamiltonian cycle - was called a "voyage around the world".

In 1930 the mathematician Menger proposed a new definition of curve length which turned out to have a close connection with TSP - the so-called *messenger problem.* Menger defined the length of a curve as the least upper bound of the set of all numbers that could be obtained by taking each finite set of points of the curve and determining the length of the shortest polygonal graph joining all points. M.M. Flood mentions (see Flood, 1955) that Hassler Whitney - the creator of matroids - posed the problem in a seminar talk at Princeton University in 1934. However, it was not until the arrival

of the computer in the late nineteen forties and early fifties, and the creation of the simplex method of Dantzig, that TSP came into real focus among the mathematicians of the world, notably through the article by Dantzig, Fulkerson and Johnson, 1954. Since then a huge amount of articles, applications, and methods concerning TSP and related topics have been published and studied. As a curiosity, Little et alt., (1963) mention that the TSP even some times is used in contests and even has achieved some public prominence. A soap company used the TSP as the basis of a promotional contest. Prizes up to $10,000 were offered for identifying the most correct links in a given 33-city problem. Quite a few people found the best cycle. On the other hand, a number of people- as the authors write – " perhaps a little over-educated, wrote to the company that the problem was impossible – an interesting misinterpretation of the state of the art"

Polynomial-time algorithms (*P*) and Non-deterministic Polynomial algorithms (*NP*)

An algorithm whose order-of-magnitude time performance is bound from above by a polynomial function of N, where N is the size of its input, is called a *Polynomial-time algorithm.* Problems where such algorithms have been shown to exist can be referred to as reasonable or tractable. Problems that in the worst case require super-polynomial- or exponential-time algorithms are referred to as unreasonable or *intractable* problems. A *verification* algorithm is an algorithm that verifies or certifies that a possible solution to a given problem really satisfies the ramifications set by the problem. *The complexity class NP* is the class of problems that can be verified by a polynomial-time algorithm. Note that $P \subseteq NP$. Given two problems P1 and P2 in the complexity class *NP*. One says that P1 is polynomial reducible to P2 if there exists a polynomial algorithm that translates problem P1 to P2. Hence, given a solution to P2, a solution to P1 can be found in polynomial time as well. A problem Q is said to be *NP-complete* if two conditions are satisfied:

1. Q is *NP*
2. All problems in *NP* must be polynomial reducible to Q

If the problem Q satisfies the second problem and not necessarily the first, Q is said to be *NP-hard.*

If a *NP*-complete problem could be solved in polynomial time, all *NP* problems could be solved in polynomial time as well. On the other hand, if any *NP* problem is intractable, every *NP*-complete problem will be intractable as well. A *NP*-hard problem can only be tractable if $P = NP$. Most researchers tend to be of the opinion that $P \neq NP$.

The TSP-*decision* problem is *NP*, since we can check whether a Hamiltonian cycle has cost less than a given upper bound in polynomial time. The TSP-decision problem is *NP*-complete and TSP is *NP*-hard. For details and further references, see for example Lawler et alt, 1985, Harel 1988, Papadimitriou and Steiglitz, 1982 or Skeina, 1998.

1.2 Notation and Definitions

Before embarking on a closer examination of different aspects of TSP, it is convenient to introduce some notation and concepts from graph theory.

In order to solve a specific instance of TSP one needs to know the costs of travelling from any city to any other city, that is, one needs to have the cost matrix for the problem. The cost matrix for a TSP will usually be denoted with capital Latin letters, *A, B, C,* and so on and the elements of a matrix with the corresponding lower-case letters with indices. Hence, a cost matrix for a TSP will be denoted $A = [a_{ij}]_{n \times n}$, where a_{ij} denotes the cost of travelling from city or node *i* to node *j,* and *n* is the number of nodes. The cost elements can always be taken to be non-negative and integer (unless some entries are irrational numbers), since we can always add the same sufficiently large number to all elements of the cost matrix without changing the optimal sequencing of the nodes, or multiply every element by some factor, to get rid of any fractions.

A *graph G* is a pair *(N,E),* where *N* is a set of points, or *nodes* and *E* is a set of *edges* connecting some or all the nodes. If there exist edges between every pair of nodes, the graph is called *complete.* To each edge in the graph, one may attach a cost. If the graph is incomplete, then we can attach an arbitrarily high cost to the non-existent edges, and in this way one may regard every graph as complete in the TSP context. The graph is called *undirected* if one can travel directly back and forth between every pair of nodes. If this is not possible, the graph is called *directed* and the directed edge is called an *arc.* Graph theoretically TSP can be described as finding the cheapest *Hamiltonian cycle* in the graph, that is a sequence of nodes and edges alternately, where every node is visited once and one returns to the starting node. If one does not return to the starting node after having visited every node, one has a Hamiltonian *path.*

To each node in an undirected graph, one associates the *degree* of the node. The degree is equal to the number of edges attached to the node. If the graph is directed, the *in-degree* of a node is the number of arcs into the node, and the *out-degree* is the number of arcs out from the node. A node in an undirected graph is called *even* if the degree is an even number and is called *odd* if the number is an odd number. A graph where all the nodes are even is called an *Eulerian graph.* By the so-called *hand-shaking* lemma there is always an even number of odd nodes in any graph or sub-graph.

A *tree* in an undirected graph is a sub-graph without cycles where every node has a degree equal or larger than 1. A node with degree 1 is called a *leave.* In a tree there is at least two leaves. By a *spanning* tree we understand a tree which includes all the nodes in the graph. Hence a Hamiltonian cycle is not a tree, but a Hamiltonian path is a special kind of a spanning tree, where exactly two nodes have degree 1 and all the others have degree 2. A *hub* is a spanning tree where all nodes except one have degree1 and the last node has degree *n - 1.*

A *Minimal Spanning Tree (MIST)* is a spanning tree with the smallest possible cost. MIST can easily be found by several different polynomial algorithms, for example the methods of either Kruskal or Prim. Deleting one edge from the optimal Hamiltonian cycle, creates a path which is a tree. This shows that MIST is a lower bound for TSP. By a *1 - tree* we understand a spanning tree plus one edge more. This will always create a graph with exactly one cycle. By a *MIST- 1-tree* we understand MIST plus the smallest edge not used in the construction of MIST. Hence, any Hamiltonian cycle is a 1-tree and MIST-1-tree is a lower bound for TSP.

In a similar fashion we define the *maximal spanning tree* (MAST) as the spanning tree with the largest possible cost. MAST can be found as easily as MIST, but note that MAST does not constitute an upper bound for the maximal Hamiltonian cycle, but only for the maximal Hamiltonian path. By a *MAST - 1 -tree* we understand the MAST plus the largest edge not used in the construction of MAST. This 1 - tree constitutes an upper bound for the maximal Hamiltonian cycle.

If the graph is directed, the concept of a tree becomes a bit more difficult. By a *semi-walk* we understand an alternating sequence of nodes and arcs $< i_1, a_1, i_2, a_2, i_3, a_3, \ldots\ldots, a_s, i_{s+1} >$ where an arc k is either the arc between i_k and i_{k+1} or i_{k+1} and i_k . A semi-walk is called a *semi-trail* if all its arcs are distinct and a *semi-path* if all its nodes are distinct and finally a *semi-cycle* if it contains at least three nodes all distinct, except that $i_1 = i_{s+1}$. Further, a directed graph is said to be *weakly connected* or *weak* if for every pair *(i,j)* of distinct nodes there exists a semi-path from *i* to *j*. Finally, with a *directed tree* we understand a weak, directed graph with no semi-cycles. A Hamiltonian path in an asymmetric matrix will then be a directed tree. A procedure to find a minimal (maximal) spanning directed tree can be found in Lawler, 1976, pp 348 - 351. An alternative approach is given in subsection 1.6.

By an *Euler trip* we understand a sequence of edges and nodes, starting in a node and ending in the same node, such that each edge is visited once. In undirected graphs this is possible iff the graph is Eulerian.

1.3 Different Categories of TSP

TSP can be categorised in different ways. Most of these categories are connected with the cost structure or the algebra of the cost matrix. Two obvious categories are to differentiate between a *symmetric* cost matrix and an *asymmetric* one, ie, if $a_{ij} = a_{ji}\ \forall i, j$, the TSP is called symmetrical (STSP) and if at least for one pair of nodes $a_{ij} \neq a_{ji}$, one says that the TSP is asymmetrical (ATSP).

Another, and maybe more important category, is the so-called *polynomial solvable cases* of TSP. These are more or less broad subclasses of TSP that can be solved by specialised algorithms, which find an optimal solution to any instance of TSP in the class, in *polynomial time.* Most of the known well-solved cases of TSP are based on

special structures in the cost matrix. Some of the classes will be treated in subsection 1.7 and some extensions will be dealt with in later chapters.

1.4 Applications of TSP

TSP is describing a rather simple situation and it may seem a bit odd that it is so hard to find an optimal solution to the problem. Due to its simplicity as a description of a practical planning situation, one should not expect to find many practical applications for the problem. However, over the years quite a few have been suggested and used, solving practical problems. A short overview of such applications is given in this section. In some of the applications it will be necessary to make small alterations to the basic problem.

In its purest form TSP can be applied to several planning situations. For example, **collecting post** from public post-boxes usually does not involve more than one vehicle, and the vehicle's capacity will usually not be an active restriction to the planning problem. The same problem arises in connection with the delivering or pick-up of cash from banks and customers by armoured cars. A slightly different planning situation occurs in **production planning**. Suppose one can produce n different commodities on a single machine, but some time or cost is incurred each time one has to prepare the machine for production of a new commodity. This set-up time and/or set-up cost will usually be sequence dependent. If one wants to minimise the set-up time or cost for producing one cycle - that is producing each of the commodities in a certain quantity, once in each cycle - one has to solve an instance of TSP. This kind of application is already mentioned in Flood, 1955. Reinelt, 1994, mentions several applications in his book. **Drilling of Printed Circuit Boards** (PCB) concerns the drilling of many holes of different diameters in order to be able to connect different layers, eg in integrated circuits. To change the diameter, the drilling equipment has to move to a toolbox. This is time consuming. Hence, all necessary wholes with the same diameter are drilled in a sequence. This is exactly a TSP situation, where one wants to minimise the time used for moving the drill from one position to another. In **X-Ray Crystallography** one has to position detectors in different places around the crystal. To move the detectors is substantially more time consuming than the actual measurements taking place. Hence, one wants to minimise the time used for moving the detectors, which turns out to be a symmetric TSP. When **overhauling Gas Turbine Engines,** one must position vanes with a lot of nozzles affixed around the circumference of the turbine. A correct placement of the vanes can give substantial benefits by reducing vibration, increasing uniformity of gas flow, and reduce fuel consumption. The problem of placing the vanes in the best possible way can be modelled as a symmetric TSP. Another application can be order-picking from a warehouse. The warehouse receivs an order for a certain subset of the commodities kept in the warehouse. Whether done manually or automatically, the **order-picking problem** - that is finding the least time consuming sequence of finding all the commodities given by the order - boils down to a TSP.

1.5 Varieties and Generalisation of TSP

There are lots of varieties and generalisations of the basic TSP. Some of these will be mentioned below.

Graphical Travelling Salesman Problem

If the underlying graph for TSP is not complete, as will be the case in many practical applications, it may well happen that Hamiltonian cycles do not exist at all in the graph. Hence, visiting all nodes, the traveller may be forced to visit the same node more than once and use some edges several times in order to reach all the nodes, and being able to return to the starting node. Then the basic assumptions for TSP are violated. However, it is reasonable to assume that one can always find a path from any node to any other node. Then by calculating the shortest paths from every node to every other node one may extend the underlying graph to become complete, and by this the graphical travelling salesman problem (GTSP) can be solved as a TSP in the extended graph. To find all the shortest paths can be done in polynomial time $O(n^3)$. To find necessary and sufficient conditions for incomplete graphs to contain Hamiltionian cycles, is in general a difficult question and will not be dealt with in this thesis, since it is outside the scope of the problems considered in the chapters to come.

The Multiple Travelling Salesman Problem

If the task of visiting the nodes in the graph is shared between two or more travellers - say m - one has what is called a multiple travelling salesman problem (m-TSP). Tacitly, one supposes that all the salesmen start from the same node and return to this node after having visiting disjointed sub-sets of the other nodes. The common starting node is usually referred to as the *depot*. The m-TSP is seemingly more difficult than TSP, but it can easily be shown that it is equivalent to a slightly larger TSP. This result was derived independently by Bellmore and Hong, 1974, Orloff, 1974, Svestka and Huckfeldt, 1973 and Eilon et alt., 1971. The technique is very simple: make $m - 1$ copies of the depot, and let the cost of travelling from one duplicate of the depot to another be infinitely large. The cost of travelling between any of the (identical) depots and one of the other nodes is the same for every copy. Travelling between the customer nodes has the same cost as before. Then one simply solves TSP with the extended cost matrix of size $(n + m - 1)x(n + m - 1)$.

K-Peripatetic Salesman Problem

In TSP one salesman is visiting every node once and in m-TSP the burden of visiting is divided between m salesmen or vehicles, but still each node receives only one visit. In K-Peripatetic Salesman Problems each node is visited by k different sales men each visiting each node only once, but travelling in such a way that the Hamiltonian cycles they use are disjoint. The problem is then to find a minimal cost of k disjoint Hamiltonian cycles. The problem was stated and named by Krarup, 1975. The term 'peripatetic' refers to a system where teachers are employed at two or more schools and travel between them. Not very much seems to be have been done with this problem, but a branch and bound algorithm for the 2-peripatetic salesman problem is

available and some well solved cases are found by de Brey and Volgenant, 1996. A practical application of the K-peripatetic TSP can be found in the area of telecommunications. The reliability of the networks increases if several cycles of communication are constructed instead of only one, especially if different cycles have no edges in common. Since TSP is a special case of the K-Peripatetic Salesman Problem - TSP corresponds to $k = 1$ - the latter is in the NP-hard class as well.

The Vehicle Routing Problem.

This is an well-known extension of the m-TSP and concerns planning situations where the customers have some kind of demand during a given time-horizon. This demand has to be delivered (or picked up) by a fleet of vehicles with restricted capacities. This so called Vehicle Routing Problem (VRP) has many applications, and a huge amount of literature exists, that discusses practical applications of the problem and the theoretical difficulties solving it to optimality. Overviews can be found in Bodin et alt., 1983, Golden and Assad (ed.), 1988, and several chapters in Ball, Magnanti et alt. (ed.), 1995b. Many extensions can be done to the basic VRP. Different models exist, depending on whether the fleet size is a á priori decision or should be a part of the decision problem. Other questions to be coped with are whether the vehicle fleet can be regarded as uniform with respect to capacities or other features related to the vehicle-customer relationship. If they cannot be regarded as uniform, one usually speaks about a non-uniform vehicle fleet. What kinds of services are performed by the vehicles? Is it a pure delivery problem, a pure pick-up problem or is it a mix of the two types of demand? In the latter case, the models will in general become more complex and difficult to solve. The basic model can be extended to incorporate several depots - VRP with multiple depots (VRPMD). If this is the case, one has to decide whether the vehicles should be allowed to return to any depot, or whether they have to return to the depot they started from. Time windows - especially in the retail business - have become more and more important. Such sophistication can be integrated in MIP models known as Vehicle Routing with Time Windows (VRPTW). One can operate with so-called hard time windows, ie time intervals that must be observed. A vehicle arriving ahead of such a time window must wait in order to perform its service. If late, then the service cannot be performed. Soft time windows can be violated, ie. the time window can be extended, but usually a penalty is incurred.

The Assignment Problem

This is the problem of finding feasible solutions of connecting nodes to each other, such that there is only one arc leaving each node and one arc entering each node. This is a feature of a Hamiltonian cycle as well, but in the present problem one does allow sub-cycles. Hence, the assignment problem (AS) is a relaxation of TSP and the optimal assignment solution constitutes a lower bound for TSP. One may of course have several non-optimal assignments as lower bounds for TSP and still other assignments larger than TSP. However, to find the optimal assignment can be done by LP, since the constraint matrix of an assignment problem is totally unimodular. In the case of a symmetric cost matrix the assignment problem can be reformulated as the finding the best solution when exactly two edges are attached to each node.

The Quadratic Assignment Problem

This problem was originally stated as a model for plant location by Koopman and Beckman, 1957. Other applications have been done in hospital planning and in typewriter keyboard optimisation.

In general the QAP involves finding the minimal sum of $\sum_{i \in N} \sum_{p \in N} d_{i\varphi(i)p\varphi(p)}$ where φ is an assignment. Often the coefficients are split into a product by stating $d_{ijpq} = a_{ip} b_{jq}$. The QAP is then called the *Koopmans-Beckmann* problem.

QAP can be shown to be a generalisation of TSP, namely by setting the matrix A equal to the cost matrix for the TSP and let the matrix B be a cyclic permutation matrix.

Hamiltonian Paths

A Hamiltonian path is - as mentioned above - a path which spans all the nodes in the graph, that is a Hamiltonian cycle minus one edge or arc. Different situations related to finding such paths will be:

i) the starting and terminal nodes s and t, respectively, are given á priori
ii) the starting (terminating) node is known, but not the terminating (starting) node, and finally
iii) neither the starting nor the terminating nodes are known.

In any of these cases the problem can be transformed to a standard TSP. In the first case one has two options. The first option is to assign a sufficiently large negative value $-M$ to the arc (t,s). Solving TSP with the slightly changed cost matrix, the optimal solution will certainly contain this arc. The second option is to enhance the graph by adding a new node, say $i = n+1$, and defining the cost for the edges $(s,n+1)$ and $(t,n+1)$ to be zero. Each Hamiltonian tour in this new graph will correspond to a Hamiltonian tour in the old one with the same cost.

In the second case, assuming that the starting node is known, one enhances the graph with a new node and add edges from all nodes in the old graph, except from s with zero cost. Solving the Hamiltonian path problem with a fixed starting node s and terminating node $n+1$ in the new graph, gives the optimal Hamiltonian path in the old with starting point s. The situation where only the terminating node is known is dealt with in a similar way. Finally, in the last case one extends again the graph with a new node, and adds edges from all nodes to the new node, all with costs zero. In this new graph, one solves the TSP.

As can be seen from the above modifications, to find the optimal Hamiltonian path is as difficult as TSP.

Bottle-neck TSP

This problem is defined as finding Hamiltonian cycles, where the cost of the most expensive edge is as small as possible, whatever the cost of the Hamiltonian cycle. Otherwise the problem is the same as an ordinary TSP. Note that an optimal solution to a bottle-neck TSP does not depend on the magnitude of the cost elements, but only on their relative values. This means, sequencing the cost elements in an increasing order and changing the values without changing the sequence, will yield the same solution for the problem. Hence, given that all the elements are different, the transformation $c'_{ij} = 2^{c_{ij}}$ will give the same solution as the original cost elements. Moreover, the transformation above shows that the bottle-neck TSP can now be solved as an ordinary TSP. Let $c'_{ij} = 2^{c_{ij}}$ be the longest arc in the optimal solution φ of TSP and suppose there exists a shorter longest arc $c'_{kl} = 2^{c_{kl}}$ in an other Hamiltonian cycle ψ. Then clearly $C'(\psi) < C'(\varphi)$, which is a contradiction. Usually, to solve a bottle-neck problem is easier than solving a TSP. Further, the bottle-neck TSP can be readily addressed as a minor variant of TSP, thus solving TSP with the added restriction that $c_{ij} \leq t$ for some value t. Hence by applying binary search, the optimal solution to the bottle-neck TSP can be found by solving *O(logn)* Hamiltonian cycle problems.

Time-dependent TSP

In this generalisation of TSP, one supposes that the cost of travelling will vary from one time period to another, the time horizon being divided into n time intervals. A linear formulation of this problem can be done with n^3 0/1-variables, and a surprisingly small amount of restrictions.

The Stacker Crane Problem

In this problem one has a pre-specified subset A of the edge set E. The objective is now to find the shortest Hamiltonian cycle containing all the arcs in A, possibly visiting some nodes several times. The problem can be applied in vehicle routing, where one has to perform pick-ups *and* deliveries where one pick-up demand occupies the capacity of the whole vehicle and is to be delivered as a total to another node.

The Travelling Purchaser Problem

In this case we have a set of nodes as in the ordinary TSP. In addition we have a set of commodities. Every commodity is available on at least one of the nodes and may be available at every node, but the price can be different from one node or market to another. Starting from a given depot, where none of the commodities are available, the traveller is supposed to find a cycle - not necessarily a Hamiltonian one - starting and ending at the depot – enabling one to buy every commodity and that the sum of the purchasing cost and travelling cost is minimised. Details can be found in Golden et alt., 1981.

The Prize Collecting TSP (PCTSP)

In this case one has node weights w_i representing some kind of benefit or income received when visiting node *i*. Starting from a chosen depot *d* with node weight $w_d = 0$, the traveller shall visit a set of the nodes once and return to the depot. The objective is to maximise profit, ie the sum of the benefits received minus the travelling costs. Note that if one assumes that all the nodes shall be visited, the sum of the benefits is given and the problem will become an ordinary TSP. However, if it is a part of the problem to decide which of the nodes to visit in order to maximise the net profit, one has a slightly different situation than TSP. PCTSP can now be solved in two different ways. The first is to maximise the profit, by finding a cycle - either Hamiltonian or a sub-cycle - that contains the depot. Note that no other sub-cycles are allowed. The second approach is to get rid of the node weights by substituting the edge weights c_{ij} with $c_{ij} - \frac{1}{2}(w_i + w_j)$, and find the cheapest cycle in the graph, containing the depot node.

Variations of this problem could be that there exists some restrictions on the time available for visiting nodes, called the *time constrained* TSP by Golden et alt., 1981, or a budget limit for the cost of visiting nodes, sometimes called the *orienteering problem* or *generalised TSP.* For details and further references, see eg. Balas (1989), Laporte and Martello, 1990 and Ramesh and Brown, 1991. The first one also contains descriptions of the relevant polytope. The second denotes the orienteering problem as the *Selective Travelling Salesman Problem.* Applications of the orienteering problem can be found in routing of oil tankers to service ships in different positions, and to maximise profit in certain production planning situations subject to time constraints.

The Covering Tour Problem

This problem was introduced by Current in 1981 and is formulated in Current and Schilling, 1989, and is further developed by Gendreau et alt., 1995. The nodes in an undirected graph are divided into two sets, *V* and *W.* The nodes in *V* are such that they *can* be visited, but they do not have to be visited. A subset *T* of *V* contains the nodes that *must* be visited. A certain node, say 1, is denoted as the depot node. The depot node is a member of *T.* A node in *W* is said to be *covered* by a tour among all nodes in *T* and possibly some nodes in *V* and not in *T*, if the distance from some node in the tour to the node in *W* does not exceed a certain distance *c.*

One application of the covering tour problem will be a transportation network where the tour corresponds to a primary vehicle route, and all points not on the tour are within easy reach from it. Similarly, another example will be the construction of routes for visiting health teams, where medical services are delivered only to a subset of places, but where all other places must also be within reasonably distances from the stopping places of the health teams. The covering tour problem is sometimes referred to as *The Travelling Circus Problem* or *Generalised Travelling Salesman Problem.* The latter name is used by Revelle and laporte (1996).

If we let $T = V = W = N$, the covering tour problem reduces to an ordinary TSP. Hence, the covering tour problem is a generalisation of TSP and is NP-hard.

The Rural Postman Problem (RPP)

In TSP one visits all nodes once and returns to the starting node. A different planning situation occurs when one in a given graph wants to visit every edge once and returns to the starting node. This is what usually is called the **Postman Problem (PP),** and was solved by Euler in 1736 for so-called *Eulerian-graphs*. Euler proved this by constructing a polynomial time algorithm that gives a sequence of nodes and edges, no edge occurring more than once in the sequence. Euler extended his algorithm to incorporate graphs with two odd nodes and relaxing the problem to a postman starting in one of the odd nodes and ending in the other, but still visiting all the edges only once. However, if the graph contains more than two odd nodes, neither Euler's first nor second approach can be applied. In 1962, the Chinese mathematician Guan Mei-gu formulated the problem of how a postman should travel in non-Eulerian graphs, such that every edge is visited alt least once and that the cost of travelling becomes as small as possible. This problem, which has become to be known as the **Chinese Postman Problem (CCP)** was solved by Edmonds and Johnson, 1973, using so-called matching techniques, leading to a polynomial algorithm for the problem.

RPP is a generalisation of CCP. Let F be a subset of the edge set E in the underlying graph. RPP is then to find the minimal closed walk in the graph visiting all the edges in F. If $F = E$, the RRP becomes CPP.

Some further variations of the TSP can be found in Noschang, 1999.

Final remark

Above some variations and applications of TSP have been briefly discussed. In many cases it turns out that the problem can be formulated as a symmetric TSP or at least closely related to this clear cut and basic formulation. However, such transformation will in most cases cost something, eg computation time or other difficulties and should be used with care before actually trying to use them in practical problem solving. For example, reformulating the GTSP as a TSP takes time $O(n^3)$, which may be unacceptable in practice. Other transformations make use of the "big M" technique, which can lead to numerical problems particularly when LP is used as a part of the solution approach. Using "big M" can also prevent finding feasible solutions to the problem when heuristics are applied.

1.6 Solution Techniques

Over the years, several solution techniques have been proposed and used for solving TSP, and related combinatorial optimisation problems. Such techniques include for example heuristics, which do not guarantee optimal solutions and mathematical programming models. In this sub-section we give a brief survey of mathematical models for TSP and some of the more well-known heuristics for this problem.

1.6.1 Exact Models for TSP

It is convenient to introduce the two linear models. Let $G = (E,N)$ be a complete graph. If the graph is undirected, let $E(i)$ denote the set of edges e, which is connected to node i. We define two sets of decision variables:

$$U_{ij} = \begin{cases} 1 \text{ if arc i - j is used} \\ 0 \text{ otherwise} \end{cases}$$

$$U_e = \begin{cases} 1 \text{ if edge e is used} \\ 0 \text{ otherwise} \end{cases}$$

The first set of variables can be used both in ATSP and STSP, and the second only in the STPS.

Model 1

$$(1.1) \qquad \min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} U_{ij}$$

st

$$(1.2) \qquad \sum_{j=1}^{n} U_{ij} = 1 \quad \forall i = 1,2,\ldots,n$$

$$(1.3) \qquad \sum_{i=1}^{n} U_{ij} = 1 \quad \forall j = 1,2,\ldots,n$$

(1.4) Subtour eliminating constraints

Model 1 can be used for ATSP as well as for symmetric TSP.

Model 2

This model can only be used in the symmetric case and usually gives better lower bounds as a starting point, and it will usually be necessary with fewer sub-tour eliminating constraints to reach optimum.

$$(1.5) \qquad \min \sum_{e \in E} c_e U_e$$

ST

$$(1.6) \qquad \sum_{e \in E(i)} U_e = 2 \quad \forall i = 1,2,\ldots,n$$

(1.7) subtour eliminating constraints

where c_e denotes the cost of edge e.

The sub-tour eliminating constraints can be formulated in many different ways. A fairly common way to formulate such constraints is to introduce inequalities of the type described in (1.8).

$$\sum_{i,j\in S} U_{ij} \leq |S| - 1 \tag{1.8}$$

where $S \subset N, 2 \leq |S| \leq n-2$.

Alternatives may be the so-called connectivity constraints (1.9)

$$\sum_{i\in S}\sum_{j\in \bar{S}} U_{ij} \geq 1 \tag{1.9}$$

where $\bar{S}$ denotes the complement of *S*. It is easily shown that (1.8) and (1.9) are equivalent.

Different ways and more compact ways of formulating sub-tour eliminating constraints have been proposed by Miller, Tucker, and Zemlin, 1960. Desrochers and Laporte, 1991, have strengthened the formulation of Miller et alt. Different approaches have been used by others, as for example Wong, 1980.

Model 1 or 2 can either be solved as IP or as LP. However, using LP will in almost all cases sooner or later give lower bounds with fractional values for the decision variables. If one wants to go on solving the given TSP as LP, one has to introduce so-called facets. Facets are valid inequalities destroying the present non-feasible solution. Facets are problem specific, and there exist a lot of them both for ATSP and STSP. Overviews of facets for TSP can be found many places, for example Laporte, 1992, Lawler et alt.,1985, and Nemhauser and Wolsey, 1988.

Asymmetric TSP can be solved as symmetric TSP by transforming the cost matrix of the asymmetric instance and increasing the number of nodes, see Reinelt, 1994, in the following way:

Let *G =(N,A)* be the graph describing the asymmetric TSP and c_{ij} be the cost of travelling along the arc *(i,j)*, $A \subseteq NxN$. We extend the node set *N* to $V = N \cup \{n+1, n+2, ..., 2n\}$ and define the undirected graph *H = (V,E)* where $E = \{(i, n+i) | i = 1,2,.....,n\} \cup \{(n+i, j) | (i, j) \in A\}$. The symmetric cost matrix corresponding to the undirected graph *(V,E)* is defined as

$$d_{i,n+i} = -M \text{ for } i = 1,2,....,n$$
$$d_{n+i,j} = c_{ij} \text{ for } (i, j) \in A$$

where *M* is a sufficiently large number, for instance the sum of all the cost elements in the original asymmetric matrix. By construction, each Hamiltonian cycle in *G* with cost D_G corresponds to a Hamiltonian cycle in *H* with cost $C_H = D_G - nM$. In

addition, an optimal cycle in H contains all edges with cost $-M$, and hence induces a directed Hamiltonian cycle in G. Using the cost matrix D we can always use model 2 above for finding the optimal solution for any kind of TSP.

1.6.2 Lower Bounds for TSP

In many cases it will be convenient to have good lower bounds for TSP. Such lower bounds can be obtained in many different ways. Any LP-relaxation to one of the models above, with or without sub-tour eliminating constraints, will give such lower bounds. Some further alternatives are reviewed in this sub-section.

1.6.2.1 Assignment Problem

An assignment solution to a given instance of TSP – that is model 1 or 2 minus constraints (1.4) or (1.7) - will give such lower bounds. The assignment problem has the so-called integrality property, that is, the problem can be relaxed to LP and we still get integer values for the decision variables. Hence, it will be very easy to find the lower bound corresponding to the minimal assignment. Assignment problems are special cases of three different combinatorial optimisation problems, namely flows in network, mathcing problems and matroid intersection problems. Extensions of the assignment problem can be done for example by allowing multi-dimensional problems. Such problems play a rôle in time table constructions, see Burkhardt, 1979.

1.6.2.2 Lagrangean Relaxation

Lagrangean relaxation was developed in the early 1970's with the pioneering work of Held and Karp, 1970 and 1971. Their work dealt specifically with TSP, but today the technique they developed is indispensable for generating lower bounds for use in algorithms to solve combinatorial optimisation problems. Relaxing the assignment equations either in model 1 or in model 2, gives a Lagrangean relaxation where the Lagrangean multipliers can have any values, and a solution to the Lagrangean optimisation problem that is also feasible to the TSP, gives the optimal solution. Whether using the sub-gradient method or multiplier adjustment to decide values for the multiplier, will usually and rather quickly give good lower bounds for the TSP and possibly reveal the optimal solution as well, if luck will have it.

1.6.2.3 Lower bounds based on trees.

Other ways of obtaining lower bounds can be with the help of sub-graphs as trees or so called 1-trees. Two such lower bounds are shown in this section and other bounds will be dealt with in the next chapter.

Shortest spanning r-arborescence

A Hamiltonian cycle in a directed graph minus one arc constitutes a directed tree, where the in-degree of each node is exactly one. Hence, a minimal spanning directed tree with an in-degree of exactly one, and where each node can be reached from the root node r is a lower bound for ATSP. The shortest of the latter structures is usually

named the shortest spanning r-aborescence. This can be found by the following model:

$$(1.10) \qquad \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}X_{ij}$$

ST

$$(1.11) \qquad \sum_{\substack{i=1 \\ i\neq j}}^{n} X_{ij} = 1 \quad \forall j$$

$$(1.12) \qquad \sum_{i\in S}\sum_{j\notin S} X_{ij} \geq 1, S \subset E; r \in S$$

$$(1.13) \qquad X_{ij} \geq 0 \quad \forall i, j, i \neq j$$

Shortest spanning 1-tree (in the symmetric case)

In the symmetric case we obtain a similar lower bound for any STSP by relaxing (and rewriting) model 2 into the following model:

$$(1.14) \qquad \sum_{i=1}^{n}\sum_{\substack{j=1 \\ i<j}}^{n} c_{ij}X_{ij}$$

ST

$$(1.15) \qquad \sum_{i=1}^{n}\sum_{\substack{j=1 \\ i<j}}^{n} X_{ij} = n \quad \forall j$$

$$(1.16) \qquad \sum_{j=2}^{n} X_{1j} = n$$

$$(1.17) \qquad \sum_{i\in S\setminus\{1\}}\sum_{j\notin S\setminus\{1\}} X_{ij} \geq 1, S \subset E\setminus\{1\}, 1 \leq |S| \leq n-1$$

$$(1.18) \qquad X_{ij} \in \{0,1\} \quad \forall i, j$$

1.6.3 Heuristics for TSP

There is a vast amount of more or less sophisticated heuristics for TSP. In this sub-section a short review of some of the most common is given. Some of the heuristics will be considered in more detail in the next chapter. For detailed overviews, see eg Bodin et alt., 1983, Reinelt, 1994 or Ball et alt., 1995a.

1.6.3.1 Construction Heuristics for TSP

These kinds of heuristics build a Hamiltonian cycle step by step. They often start with a single node chosen randomly and extend a path or a sub-cycle node by node, until a Hamiltonian cycle is found. Each of the heuristics can often be performed many times for the same instance of TSP, due to the randomness of the starting node. They also often come in slight variations of the basic idea. These variations are not treated

extensively below. Most of the heuristics below are originally created for STSP, but many of them can be applied on ATSP, usually at the cost of some more calculations.

Nearest neighbour (NN)

This is perhaps the simplest of all heuristics for TSP, and is a greedy heuristic trying to connect nodes closest to each other in a sequential way. The standard version can be formulated as follows:

Step 1: Select an arbitrary node j and let $W = \{1,2,.....,n\} \setminus \{j\}$, set $l = j$.
Step 2: As long as $W \neq \emptyset$, do the following:

2.1 Let $j \in W$ such that $c_{lj} = \min\{c_{li} | i \in W\}$
2.2 Connect l to j and set $W = W \setminus \{j\}$ and $l = j$

Step 3: Connect l to the node selected in step 1 to form a Hamiltonian cycle.

The heuristic creates a path, which in the end is closed to create a Hamiltonian cycle. A simple alternative version of this, is to let the current path be extended in both its end nodes. The heuristic can be applied on both ATSP and STSP. Obviously, the last step can incur a high cost.

Insertion heuristics

The basic idea of this family of heuristics is to start with a sub-cycle. The number of nodes in this sub-cycle can be two (and even a loop of one node), or many, but not all. One then tries to extend this sub-cycle by adding one new node at a time and inserting the chosen new node into the existing sub-cycle by some criterion and hence creating a new and larger sub-cycle. Formalised, the procedure can be described as follows:

Step 1:

Select a starting sub-cycle

$i_1 - i_2 - i_3 - ..... i_k - i_1$ and let $W = \{1,2,.....,n\} \setminus \{i_1, i_2, ..., i_k\}$

Step 2: As long as $W \neq \emptyset$, do as follows:

2.1 Select a node $j \in W$ according to some criterion
2.2 Insert node j at some position in the existing sub-cycle by some criterion and set $W = W \setminus \{j\}$

Due to the "openness" of step 2.1 and 2.2 many different versions exist for this basic idea. Usually, the criterion used in step 2.2 is to insert the chosen node in a position such that the increase in cost will be as small as possible. Such a criterion will always incur some calculations performed k times, if the existing sub-cycles have k edges or

arcs. A node, which is a part of the existing sub-cycle, is called a *tour node.* It is convenient to make the following definitions:

For $j \in W$ one defines $d_{\min}(j) = \min\{c_{ij} | i \in N \setminus W\}$ which is the smallest distance from any tour node to any node not in the existing sub-cycle. In a similar way, the maximal distance from any tour node to any node not in the existing sub-cycle is defined by $\forall j, j \in W,\ d_{\max}(j) = \max\{c_{ij} | i \in N \setminus W\}$. Further, the sum of all distances from a tour node to all nodes not in the existing sub-cycle is denoted *s(j)*, that is, $s(j) = \sum_{i \in N \setminus W} c_{ij}$.

Nearest insertion

Insert the node k that has the shortest distance to any tour node, that is choose k such that $k \in W$ and $d_{\min}(k) = \min\{d_{\min}(l) | l \in W\}$

Farthest insertion 1

Insert the node k not in the sub-tour, such that the minimal distance to the tour is maximal. That is, choose k such that $k \in W$ and $d_{\min}(k) = \max\{d_{\min}(l) | l \in W\}$.

Farthest insertion 2

Insert the node k not in the sub-tour with the farthest distance to the tour. That is, choose k such that $k \in W$ and $d_{\min}(k) = \max\{d_{\max}(l) | l \in W\}$.

Farthest insertion 3

Insert the node k not in the sub-tour, such that the maximal distance to the tour is minimal. That is, choose k such that $k \in W$ and $d_{\max}(k) = \min\{d_{\max}(l) | l \in W\}$.

Cheapest insertion

In the insertion procedures above, different selecting criteria are specified. These criteria exclude all but one single node not in the present sub-tour. Then one has to calculate where to insert the selected node in the existing sub-tour, choosing the place such that the increase in cost is as small as possible. In cheapest insertion one calculates the increased cost for all nodes not in the existing sub-tour, and chooses the node and insertion with the smallest cost increase. Hence, the calculations involved are many and take time, and there exists variants of the cheapest insertion heuristic, which do not consider all possibilities, but selects some. However, these variants will not play any important part in this thesis and will therefore not be discussed further.

Random insertion

Here, the new node, which is going to be inserted into the existing sub-tour, is chosen at random.

Largest sum insertion

Insert the node k not in the sub-tour, such that the sum of the distances to the tour is maximal. That is, choose k such that $k \in W$ and $s(k) = \max\{s(l)|l \in W\}$.

Smallest sum insertion

Insert the node k not in the sub-tour, such that the sum of the distances to the tour is minimal. That is, choose k such that $k \in W$ and $s(k) = \min\{s(l)|l \in W\}$.

Fast Versions of Insertion Heuristics

Insertion heuristics involve two kinds of calculations. One type of calculation is needed when applying the selecting criterion and one when the insertion criterion is applied. In order to avoid too many calculations, one can simplify one or both of the two calculations, egv by making á priori decisions about where to insert, or restricting the search for the new node to a part of the remaining nodes. For details about these kinds of variants, see Reinelt, 1994 or Bodin & alt, 1983. However, these variants will not play any important part in what follows. Note that all the mentioned varieties of the insertion heuristics can be applied on both ATSP and STSP.

Convex hull

All the insertion heuristics above usually start with selecting the first node at random, and so a sub-tour is gradually built from this starting node, using whatever criteria involved. For TSPs with Euclidian cost matrices, which always are symmetrical, it is easy to construct the convex hull. This can be done in *O(nlogn).* The convex hull will either be a Hamiltonian cycle - and if so, this cycle will be the optimal solution (Folklore) - or the convex hull will be a sub-tour with some nodes in the interior of the hull. This sub-tour can then be used as a starting sub-tour for any of the insertion heuristics. Moreover, the sequence of the nodes in the convex hull will be the same as in the optimal solution (see Lawler et. alt., ch 7). Flood, see Flood, 1956, observed that "in the Euclidean plane the minimal (or optimal) tour does not intersect itself".

Other possibilities exist as well when starting with a convex hull. These heuristics basically try to exploit geometrical features of the node map, due to the underlying Euclidian metric. Specifically, one can mention *Convex hull insertion procedure,* Stewart, 1977, *Greatest angle insertion procedure,* Norback & Love, 1977 and1979 and *Ratio times difference insertion procedure,* Or, 1976, all of which can be found in Lawler & alt, 1985, chapter 7, p. 217. Again, these heuristics will not play any important part in what follows, so the details are left out.

The Saving Heuristic

This heuristic was originally made for VRP, see Clarke and Wright, 1963, but can easily be adapted to the more simple case of TSP.

The heuristic is based on calculation of the so-called *savings*. In the TSP case one chooses one node, say *d,* as a depot. Visiting two other nodes separately creates a certain cost. This cost is compared with the cost incurred when the two nodes are visited sequentially and the difference is the so-called saving value for the pair of nodes relatively to the chosen depot node. The saving values s_{ij}^{d} can then be calculated by the formula below, see also figure 1.3a and b

$$s_{ij}^{d} = c_{id} + c_{dj} - c_{ij}$$

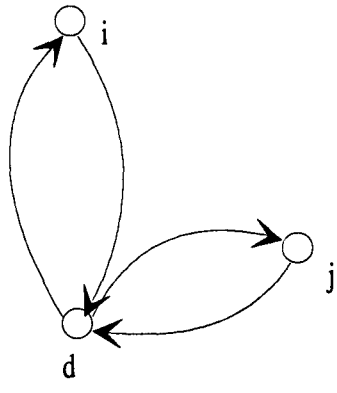


fig. 1.3a

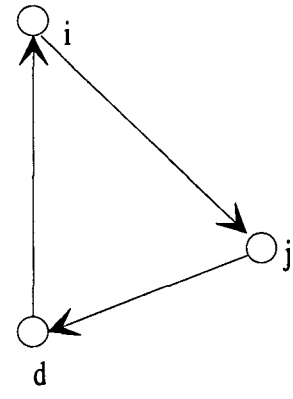


fig.1.3b

When the savings are calculated for a given depot, one sorts the savings in decreasing order and forms a Hamiltonian path among all the nodes, except the depot. This path is created in a greedy fashion, choosing the arc with the largest saving value not used so far, and deleting all savings that create sub-cycles or nodes with out-degree or in-degree greater than one. When all nodes have been taken care of, the path is closed by connecting the end nodes to the depot.

Hence, the saving heuristic is more or less the same type of heuristic as the nearest neighbour heuristic above, but performed with a different cost matrix than the original one. Note also that the saving values are only used for creating some sequence of the nodes, hoping against hope that this sequence will give a small cost when measured against the original cost matrix. Usually one does not even bother to calculate the sum of the applied savings. The saving heuristic can be applied on both ATSP and STSP. However, the transformation done by calculating the saving values turns out to have no effect on the sequence of the different Hamiltonian cycles, see below, other than reversing the sequence. This fact was already observed by Flood, 1956, who basically formulates the same transformation as Clarke and Wright used, some years later. There is no reference in their article to Flood's observation.

The loss heuristic

All the above heuristics are greedy in some sense, and it is not difficult to find examples of instances of TSP were these heuristics do not lead to the optimal solution, showing that these greedy policies are not necessarily the best ones. The loss heuristic tries to avoid this by not taking the "nearest" node.

The loss heuristic was first proposed by Webb, 1977, for symmetric matrices, and was later extended to asymmetric TSPs by Van der Cruyssen and Rijckaert, 1978. The core idea of the heuristic is to calculate the lost opportunity cost of not linking a city to its nearest neighbour.

Starting with some path, one out of four categories must occur for each node in the graph. These categories are shown in the table below:

Category	A	B	C	D
In-degree	1	1	0	0
Out-degree	1	0	1	0

A node in category A is not an end point of the path and is not pursued any further because both the arc into the node and out from the node have already been decided, and this decision is never changed.

If a node j belongs to category B, one has the situation in fig 1.4

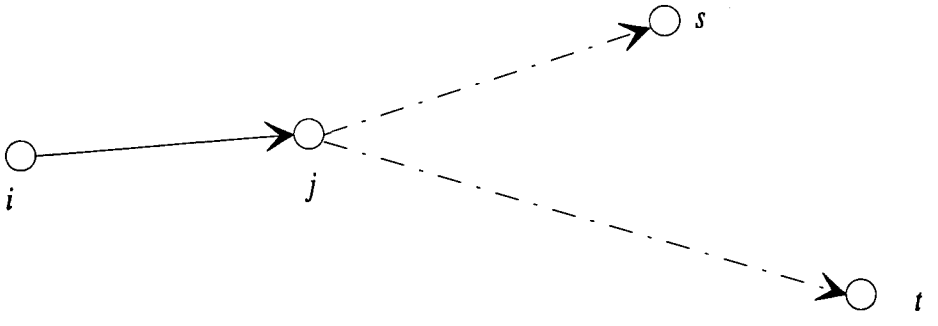
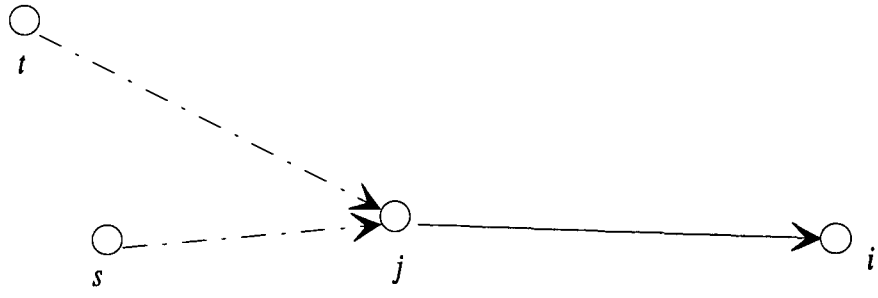


fig. 1.4

Note that since the in-degree is one, the predecessor i of j is already decided. In figure 1.4, s is supposed to be the nearest neighbour of j, and t the next nearest, such that no sub-cycle is formed. The lost opportunity of not linking j to s is then calculated by

$$LOSS(j) = c_{jt} - c_{js}$$

In a similar fashion the category C is treated, but in this case the successor of j is already decided, since the out-degree is one. The situation is described in fig.1.5, where the nodes s and t have the same properties as in the previous case.



fig,1.5

The lost opportunity cost is then calculated by

$$LOSS(j) = c_{tj} - c_{st}$$

The last category is treating the situation where the node *j* has no connection with the existing path, that is, there are no arcs into or out from the node in question. The situation is described in figure 1.6.

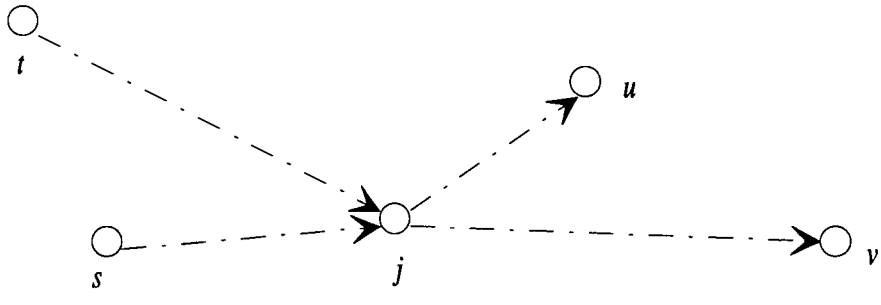


fig.1.6

If all five nodes are different, the cost of lost opportunity is calculated as

$$LOSS(j) = \max\left\{c_{tj} - c_{st}, c_{jv} - c_{ju}\right\}$$

At each step in the procedure, we calculate *LOSS(j)* for all nodes that fall into categories B, C and D. The node with the largest loss is then connected to its most favourable neighbour. The heuristic can be applied on both ATSP and STSP.

Heuristics based on minimal spanning trees

For any symmetric instance of TSP the minimal spanning tree is easily found, and the value of MIST is a lower bound for TSP. If MIST is a path, we have found the optimal solution to the problem of finding the cheapest Hamiltonian path, but in general, the degree of some of the nodes in MIST will be larger than two. However, it is still possible to hope that at least some of the edges contained in MIST will be a part of the optimal solution of TSP. Below, two heuristics based on this hope are presented. Since finding a directed MIST in an asymmetrical matrix is a bit complicated - but still polynomial - this kind of heuristic is usually only applied on STSP.

Doubling of the minimal spanning tree

Construct MIST from the given cost matrix. Then, double each edge in the tree. The resulting graph then becomes an Eulerian graph since every node has an even degree. In this graph it is possible to construct many Eulerian tours. Construct one such tour. Then starting from any node in the Eulerian tour, follow this tour edge to edge until a node is reached a second time. Then retreat to the previous node in the Eulerian tour and shortcut - ie use an edge not in MIST - from this to the first node in the Eulerian tour which has not yet been visited. Proceed like this until all nodes are visited and then return directly to the starting node. Note that for each choice of the Eulerian tour we can start the procedure in n different ways. Hence, the heuristic can be performed in a variety of ways, generating very many Hamiltonian cycles.

Christofides heuristic

This heuristic is a variant of the previous one starting in the same way by finding MIST. Since every graph contains an even number of odd nodes by the so-called "handshaking" lemma, the odd nodes in MIST can be matched in many ways. Finding the minimal matching in MIST, an Eulerian graph is created. From this Eulerian graph, we proceed as in the previous heuristic.

1.6.3.2 Improvement Heuristics for TSP

The best known heuristics of this kind, are the so-called edge change procedures, see Lin, 1965 and Lin and Kernighan, 1973. These heuristics start with a given Hamiltonian cycle of some cost. Then two or more edges or arcs are removed from the cycle. These edges or arcs are replaced by other edges or arcs, such that new Hamiltonian cycles are formed. The costs of the new cycles are then compared with the old cycle. If k different edges are removed from a cycle, the change is called a *k-change*. If all k-changes starting from a certain cycle are tried, then the best Hamiltonian cycle is called a *k-optimum* or this specific new cycle is *k-optimal.* Usually the heuristic will perform better and better as the number k is increased, but on the other hand, the number of new cycles that can be made after the removal of k edges, will increase very rapidly as will the number of different ways to select the k edges. Hence, this kind of improvement heuristics is usually restricted to values of $k = 2$ or $k = 3$. Improvement heuristics will usually perform well, but in the context of this thesis nothing new will be offered. Consequenyly, no further details seem to be necessary. The heuristic can be applied on both ATSP and STSP.

1.6.3.3 Composite Heuristics for TSP

This kind of heuristic is just a combination of any construction heuristic and an improvement heuristic. That is, we apply a construction heuristic in order to get a Hamiltonian cycle, and use this Hamiltonian cycle as input to the improvement heuristic.

1.6.3.4 Heuristics Based on Lagrangean Relaxation

Each time one solves a Lagrangean relaxation problem for a concrete TSP, one has the opportunity to change the solution of the Lagrangean problem, into a feasible solution to the concrete TSP. Then one has obtained an upper bound for the same TSP. Taken together with the lower bound created by the Lagrangean relaxation, this can give good indications of how far from the target one is.

1.6.3.5 Meta-heuristics and other Heuristics

Throughout the last decades, several so-called local-search methods have been tried in order to find good solutions to combinatorial optimisation problems, including TSP. A very brief survey of such methods is given below.

Simulated annealing

This successive improvement method is derived from an analogy with material annealing processes used in mechanics, and was introduced by Kilpatrick et alt in 1983. In order to bring a material to a minimal-energy solid state, it is necessary to heat it until the particles are randomly distributed in the liquid state. Then to avoid local minima, the temperature is gradually reduced until the system reaches an equilibrium. At a high temperature, all possible states can be reached, but as the system cools down the number of possibilities is reduced, and the process converges to a stable state. Applied to combinatorial optimisation, the aim is to move from a given initial solution to minimum-cost solution, by performing gradual changes from the starting solution. In the beginning, the cost is high and the number of allowed moves or changes is high as well. Changes leading to smaller costs, decrease the number of allowed changes as well, until no further change is possible. For a given starting value, the simulated annealing algorithm is identical with the *k-opt* procedure, calculating all solution in the neighbourhood of the initial solution. Sometimes new solutions with higher costs are allowed, in order to reduce the probability of becoming trapped in a local optimum. Simulating annealing has been applied to TSP by many authors, with apparently a mixed degree of success.

Tabu-search

This method was introduced by Glover in 1977 and is like the previous heuristic a general heuristic for combinatorial optimisation problems. Again, a feasible but not necessarily good solution, is the starting point. A neighbourhood is defined to this solution. In order to prevent cycling, solutions that have already been examined are forbidden and entered into a "tabu-list" which is constantly updated. A certain solution in the tabu-list stays there for a certain time defined by the user. The best solution in the difference between the neighbourhood and the tabu-list is then found, and the procedure starts again. Limitations are given to the number of iterations. Tabu-search has been applied to TSP with seemingly very positive results.

Genetic algorithms

Genetic algorithms are sometimes called population heuristics or bionomic heuristics. The basic idea goes back to Lawrence Fogel in 1960. Tabu search and simulated annealing work on improving a single current solution. Genetic algorithms use a number of current solutions and combine them together to generate new and hopefully, better ones. A genetic algorithm can be described as an "intelligent" probabilistic search algorithm. It basically tries to simulate the evolutionary process of biological organisms in nature. During the course of evolution, natural populations evolve according to the principles of natural selection and "survival of the fittest". Genetic algorithms maintain a "population" of feasible solution candidates for the problem. Elements are drawn at random from this population and allowed to "reproduce" by combining some aspects of two parent solutions. The probability that an element is allowed to reproduce is based on its "fitness", that is basically a function of the cost of the solution it represents. Unfit elements die from the population, and are replaced by more successful "children". The idea behind genetic algorithms is very appealing, but the convergence is slow and its success is varied. A variant of genetic algorithms combining such techniques with local search techniques is so-called **memic algorithms.**

Neural networks is still another approach to solving combinatorial optimisation problems. For a short introduction with further references, see for example Skiena, 1998. Recently Colorni and alt.,1992 has introduced so-called **Ant Systems,** using a colony of co-operating ants to solve the TSP. For further references, see eg Noschang, 1998.

A general bibliography for the TSP can be found Jünger et alt., 1997.

1.7 Polynomial Cases for TSP

The TSP seems to be notoriously difficult to solve in general for large n. However, during the last decades one has managed to identify certain sub-classes of TSP where polynomial time algorithms have been found. Good overviews of such cases can be found in Lawler & alt, 1986, ch. 4 and in Burkard & alt., 1995. Some of these classes will be briefly reviewed below in as much as they will be treated or used in the chapters to come.

1.7.1 The Constant-TSP

This is probably the simplest of the special cases of TSP and describes a cost matrix that gives the same cost for every Hamiltonian cycle in the graph. The matrix is readily described in the following theorem by Berenguer, 1979.

Theorem 1.1

The only matrices C for which all cyclic permutations on the n nodes have the same length are those which have the structure $C = [a_i + b_j]_{nxn}$.

Note, that the *a*'s and the *b*'s can be negative as well as positive numbers.

Corollary 1.1

In a constant TSP matrix, the cost of any Hamiltonian cycle is $\sum a_i + \sum b_j$ *and the costs of the shortest and longest Hamiltonian paths are*

$$\sum a_i + \sum b_j - \max_{i,j}\{a_i + b_j\} \text{ and } \sum a_i + \sum b_j - \min_{i,j}\{a_i + b_j\}$$

respectively.

By a *linear admissible transformation* we understand a transformation of the cost elements of any quadratic cost matrix, such that the total ordering of all cycles is preserved or reversed. Such linear transformation can be

$$C(\varphi) = \alpha + \beta C(\varphi)$$

where α and β are constants. The order is preserved if β is positive, and reversed if negative. The term 'admissible transformation' was introduced by Vo-Khac, 1971, and was used on transformations upon matrices, such that the order of cycles was the same before and after the transformation. Berenguer restricts this to *linear* admissible transformation and proves the following theorem:

Theorem 1.2

The only linear admissible transformations are those obtained by adding constants a_i *to the i-th row and* b_j *to the j-th column of a scalar multiple of C.*

Evidently, the set ***C*** of all matrices *nxn* where all cyclic permutations give the same cost, forms a linear sub-space of the vector space formed by all *nxn* matrices. Lenstra and Rinnoy Kan, 1979, proved the two following lemmas:

Lemma 1.1

The dimension of the vector space ***C*** *is 2n - 1.*

Let R_i (C_j) denote matrices where the *i*-th row (*j*-th column) only contain ones and all other elements are zeros.

Lemma 1.2

Any subset of 2n - 1 matrices from $\{R_1, R_2, \ldots\ldots, R_n, C_1, C_2, \ldots\ldots, C_n\}$ *forms a basis for the vector space C.*

A sub-class of the constant-TSP is the so-called *potential* matrices described by Warren, 1990.

A square matrix is called *potential* iff $p_{ij} + p_{jk} = p_{ik}$ for all triples i, j, k. Further, a square matrix is called *constant row (column)* iff all its entries in each row (column) are equal. The following theorem characterises and solves the TSP for potential matrices.

Theorem 1.3

A matrix P is potential iff P = E + F where E is a constant row matrix and F is a constant column matrix, such that the entries on the main diagonal of E + F are 0.

A potential matrix is skew-symmetric and the length of any assignment is 0.

1.7.2 The Small TSP

Let $\boldsymbol{a}$ and $\boldsymbol{b}$ be two n-dimensional vectors and C a nxn matrix. The matrix C is called *small* iff there exists two vectors such that $c_{ij} = \min\{a_i, b_j\}$. Without loss of generality and for simplicity of notation, one can always assume that $a_1 \leq a_2 \leq \ldots\ldots\ldots \leq a_n$. A small matrix is said to have *distinct values* if all the elements of $\boldsymbol{a}$ and $\boldsymbol{b}$ are distinct. Let d_i be the i-th smallest of the $2n$ distinct values of the elements of the two vectors. Further, let $D = \{d_1, d_2, \ldots\ldots, d_n\}$ and $d = \sum_{i=1}^{n} d_i$. We then have the following theorem, due to Gabovich, 1970, which also and can be found in Lawler & alt, 1985, chapter 4.

Theorem 1.4

Let C be a small matrix with distinct values. The length of an optimal tour in C is d iff one of the following three conditions holds:

a) *For the same city i, both* a_i and b_j *are in D*

b) $D = \{a_1, a_2, \ldots\ldots, a_n\}$

c) $D = \{b_1, b_2, \ldots\ldots, b_n\}$

The theorem just gives conditions for the optimal solution and the corresponding value, but is not constructive in the sense that it specifies the sequence of the nodes that leads to this value. However, the proof of the theorem is constructive and shows

how to find such a sequence. We will return to the proof in chapter 3, where an extension of this class is done. There exists several extensions of theorem 1.4 in Lawler & alt, 1985, where some of the restrictions in theorem 1.4 are removed.

1.7.3 Graded TSP

A matrix C is called *graded across its rows* iff $c_{ij} \leq c_{i,j+1}$ for all i, j. It is called *graded up its columns* iff $c_{ij} \geq c_{i+1,j}$ for all i, j and *doubly graded* iff it is graded both across its rows and up its columns. To find TSP in a graded or even doubly graded matrix is *NP-hard*, since any matrix can be made graded or doubly graded by a linear admissible transformation. Hence, it would be a big surprise if one managed to find a polynomial algorithm for graded matrices. The following theorem gives, however, a useful approximation to this kind of matrices, see Lawler & alt, 1985, ch.4.

Theorem 1.5

Let C be non-negative and graded up its columns, and let φ be an optimal assignment. Then it is easy to find a cycle τ, such that

$$C(\varphi) \leq C(\tau) \leq c(\varphi) + \max_j \{c_{1j}\}$$

Again, the theorem in itself is not constructive, but the proof is. We will return to this proof in chapter 3 where the ideas of the proof are utilised in a different context.

It turns out that if we try to solve a bottle-neck TSP for doubly graded matrices, the following theorem applies, see Lawler & alt, 1985, ch.4.

Theorem 1.6

If C is doubly graded, then a bottle-neck optimal cycle is given by the permutation 1 – 2 - 3 - .....- n-1 –n - 1

1.7.4 The Circulant TSP

A circulant matrix is a *nxn* matrix of the form where the elements in positions *(i,j)* such that *(j - i) = k (mod n)* has the same cost c_k. These elements constitute the *k*-th stripe of the matrix *C*. All stripes yield feasible assignments for *C*. Hence, finding the cheapest assignment can be done by inspection, namely the stripe with the smallest c_k. For circulant matrices we have the following theorem by Garfinkel, 1977.

Theorem 1.7

The number of sub-cycles in the assignment given by the k-th stripe is the greatest common divisor of k and n, gcd(k,n).

Theorem 1.7 gives immediately the following two corollaries.

Corollary 1.2

If gcd(k,n) = 1, then the k-th stripe yields a Hamiltonian cycles.

Corollary 1.3

If n is prime, each stripe, other than the 0-th, yields a Hamiltonian cycle.

The question concerning whether this class of matrices is polynomial or not, is still open, but restricted to finding the optimal Hamiltonian path is polynomial according to the next theorem by Bach, Luby and Goldwasser, which can be found in Lawler et alt., 1985, ch.4.

Theorem 1.8

The nearest neighbour heuristic starting from any node yields a shortest Hamiltonian path.

It is an open question whether TSP for circulant matrices is NP-hard or not. Even if the matrix is specified to be circulant and symmetric, the question remains open. However, the problem for symmetric, circulant *bottle-neck* TSPs was shown to be efficiently solvable by Burkard and Sandholzer, 1991.

1.7.5 Product Matrices

In sub-section 1.7.1 the so-called constant-TSP was considered. In this sub-class the cost of travelling from one node to another is separated into two independent parts, using "plus" as the algebra. For product matrices the separation of the cost elements is done by multiplication. Hence, let $\boldsymbol{a}$ and $\boldsymbol{b}$ be two vectors with n elements, and define the elements of a product matrix by

$$c_{ij} = a_i b_j$$

Theorem 1.9 shows that this class of matrices can be as hard to solve to optimality as TSP as such, see Sarvanov, 1980, which also can be found in Lawler et alt.,1985, ch.4.

Theorem 1.9

The TSP restricted to product matrices is NP-hard.

However, specifying the product matrices further, by craving symmetric matrices as well gives a more positive result as the following theorem shows, see Gaikov, 1980 which also can found in Lawler et alt., 1985, ch. 4. Note that a symmetric product matrix has the form

$$c_{ij} = \lambda a_i a_j$$

where λ is a constant.

Theorem 1.10

For symmetric product matrix C, where $c_{ij} = \lambda a_i a_j$ and $a_1 \leq a_2 \leq \ldots\ldots \leq a_n$, there exists a polynomial time algorithm to find an optimal Hamiltonian cycle.

Note that the extra premise that the elements of the vector ***a*** shall be ordered in an ascending sequence does not lead to any reduction of generality. The proof of theorem 1.10 comes as a corollary to a rather difficult theorem with a long and technical proof. However, it turns out that an alternative and much simpler and direct more proof is possible. This proof leans heavily on theorem 1.11 by Hardy, Littlewood and Pólya, 1934. Let *(a)* denote a set of real numbers with *n* elements. Let *(a)* be the set *(a)* rearranged in ascending order, that is, $\underline{a}_1 \leq \underline{a}_2 \leq \ldots\ldots \leq \underline{a}_n$.

Theorem 1.11

If (a) and (b) are given and are of the same cardinality and without any special ordering, then $\sum ab$ is greatest when (a) and (b) are monotonic in the same sense and least when they are monotonic in the opposite senses; that is to say

$$\sum_{i=1}^{n} \underline{a}_i \underline{b}_{n+1-i} \leq \sum_{i=1}^{n} a_i b_i \leq \sum_{i=1}^{n} \underline{a}_i \underline{b}_i$$

Before giving an alternative proof for theorem 1.10 it can be convenient to consider an example. It is necessary to discriminate between graphs where the number of nodes is even and odd. So let the underlying vector be as in theorem 1.10.

For a graph with an even number of nodes, say $n = 10$, the minimal cost is obtained as illustrated in figure 1.7.

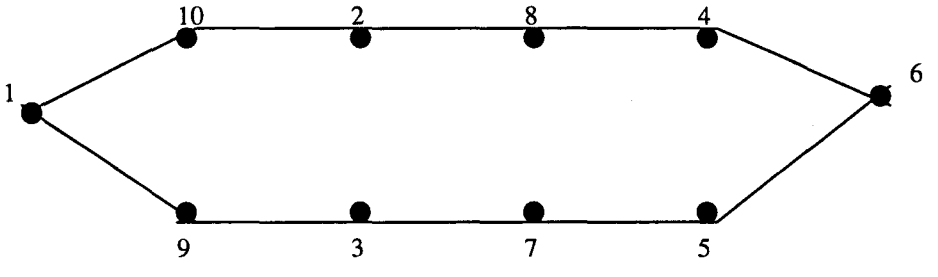


fig. 1.7

The cost of the above cycle is

$$S = a_1a_{10} + a_{10}a_2 + a_2a_8 + a_8a_4 + a_4a_6 + a_6a_5 + a_5a_7 + a_7a_3 + a_3a_9 + a_9a_1$$

Note, that in this cycle the smallest element is multiplied with the two largest ones. The second smallest is multiplied with the largest and the third largest and so on. The products in the sum above can be re-arranged in the following way

$$S = a_1a_{10} + a_2a_{10} + a_3a_9 + a_4a_8 + a_5a_7 + a_6a_5 + a_6a_4 + a_7a_3 + a_8a_2 + a_9a_1$$

The first factors in the ten products constitute the set

$$B = \{a_1, a_2, a_3, a_4, a_5, a_6, a_6, a_7, a_8, a_9\}$$

where the elements are in an increasing order.

The second factors constitute the set

$$C = \{a_{10}, a_{10}, a_9, a_8, a_7, a_5, a_4, a_3, a_2, a_1\}$$

where the elements are in decreasing order. Hence, by theorem 1.11 the sum S is the smallest possible when multiplying the elements of B and C pair ways. Note that the set B contains element no. 6 twice and element no.10 is not present in this set. In set C, element no. 6 is not present, but the largest element, no 10, is present twice.

In the case where the graph has an odd number of nodes, the situation is slightly different. Let the number of nodes be $n = 11$. In this case the minimal sum becomes as illustrated in figure 1.8

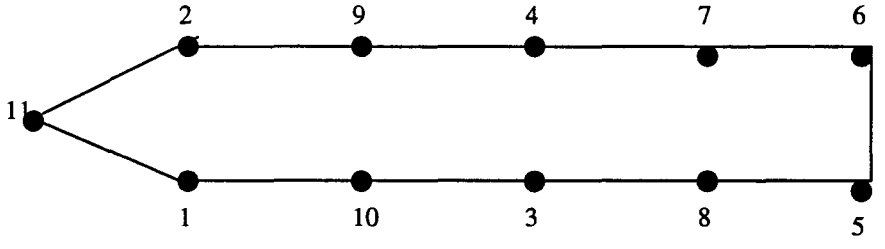


fig. 1.8

The cost of the above cycle is

$$T = a_1a_{11} + a_{11}a_2 + a_2a_9 + a_9a_4 + a_4a_7 + a_7a_6 + a_6a_5 + a_5a_8 + a_8a_3 + a_3a_{10} + a_{10}a_1$$

Note, that in this cycle the largest element is multiplied with the two smallest ones. The second largest is multiplied with the smallest and the third smallest and so on. The products in the sum above can be re-arranged in the following way

$$T = a_1a_{11} + a_2a_{11} + a_3a_{10} + a_4a_9 + a_5a_8 + a_6a_7 + a_6a_5 + a_7a_4 + a_8a_3 + a_9a_2 + a_{10}a_1$$

The first factors in the eleven products constitute the set

$$B' = \{a_1, a_2, a_3, a_4, a_5, a_6, a_6, a_7, a_8, a_9, a_{10}\}$$

where the elements are in an increasing order.

The second factors constitute the set

$$C' = \{a_{11}, a_{11}, a_{10}, a_9, a_8, a_7, a_5, a_4, a_3, a_2, a_1\}$$

where the elements are in decreasing order. Hence, by theorem 1.11 the sum S is the smallest possible when multiplying the elements of B' and C' pair ways. Note that the set B' contains element no. 6 twice and element no.11 is not present in this set. In set C', element no. 6 is not present, but the largest element, no. 11; is present twice.

Alternative proof of theorem 1.10:

Let $\boldsymbol{a}$ be the vector describing the symmetric product matrix, and as mentioned, let the numbering of the nodes be done such that the elements come forth in an ascending order. First, let the number of nodes n be an even number. Now, let S be the following sum:

$$S = a_1 a_n + a_2 a_n + a_3 a_{n-1} + \ldots\ldots + a_i a_{n+2-i} + \ldots\ldots + a_{\frac{n}{2}} a_{\frac{n}{2}+1} + a_{\frac{n}{2}+1} a_{\frac{n}{2}} + a_{\frac{n}{2}+1} a_{\frac{n}{2}-1} +$$

$$a_{\frac{n}{2}+2} a_{\frac{n}{2}-2} + \ldots\ldots\ldots + a_i a_{n-i} + \ldots\ldots + a_{n-2} a_2 + a_{n-1} a_1$$

Note, it is easily checked that the sum above corresponds to a Hamiltonian cycle. Now, consider the first factors in the sum above. These can be described as a sequence

$$b_i = \begin{cases} a_i & \text{for } i = 1,2,3,\ldots\ldots,\dfrac{n}{2}+1 \\ a_{\frac{n}{2}+1} & \text{for } i = \dfrac{n}{2}+2 \\ a_{i-1} & \text{for } i = \dfrac{n}{2}+2,\ldots\ldots,n \end{cases}$$

These numbers form an ascending sequence. The second factors in the above sum form the following sequence:

$$c_i = \begin{cases} a_n & \text{for } i = 1 \\ a_{n+2-i} & \text{for } i = 2,3,\ldots\ldots\ldots,\dfrac{n}{2} \\ a_{n+1-i} & \text{for } i = \dfrac{n}{2}+1,\ldots\ldots,n \end{cases}$$

These numbers form a descending sequence and hence by theorem 1.11 the sum S is the smallest sum obtainable from pair ways multiplication of the two sequences.

Now, let φ be any Hamiltonian cycle on the matrix $A=[a_i a_j]$. Then

$$A(\varphi) = a_1 a_{\varphi(1)} + a_2 a_{\varphi(2)} + \ldots\ldots\ldots + a_n a_{\varphi(n)}$$

Then there exists an index t such that $\varphi(t) = n$, and an index p such that $\varphi(p) = \frac{n}{2} + 1$. Switching the factors in the two products containing the indices t and p the cost of the cycle can be written

$$A(\varphi) = a_1 a_{\varphi(1)} + a_2 a_{\varphi(2)} + \ldots\ldots\ldots + a_{\frac{n}{2}+1}\, a_{\varphi(\frac{n}{2}+1)} + a_{\frac{n}{2}+1} a_p + \ldots\ldots + a_{\varphi(n)} a_n + a_{\varphi(t)} a_n$$

Since φ is a cyclic permutation, $\varphi(n) \neq \varphi(t)$. Without loss of generality we can now assume a certain order of the indices generated by the cyclic permutation, for instance

$$\varphi(n) < \frac{n}{2} + 1 < \varphi(t)$$

Then the cost of the cycle can be re-arranged as follows:

$$A(\varphi) = a_1 a_{\varphi(1)} + a_2 a_{\varphi(2)} + \ldots\ldots + a_{\varphi(n)} a_n + \ldots\ldots + a_{\frac{n}{2}+1}\, a_{\varphi(\frac{n}{2}+1)} + a_{\frac{n}{2}+1} a_p + \ldots\ldots + a_{\varphi(t)} a_n + \ldots\ldots + a_{n-1} a_{\varphi(n-1)}$$

Now, the first factors in the products in the sum above forms the sequence

$$a_1 \leq a_2 \leq \ldots\ldots \leq a_{\frac{n}{2}+1} \leq a_{\frac{n}{2}+1} \leq a_{\frac{n}{2}+2} \leq \ldots\ldots\ldots \leq a_{n-1}$$

which corresponds to the sequence b_i defined above. The second factors in the sum above consist of the following numbers in some order:

$$\left\{ a_1, a_2, \ldots\ldots, a_{\frac{n}{2}}, a_{\frac{n}{2}+2}, \ldots\ldots\ldots, a_{n-1}, a_n, a_n \right\}$$

corresponding to the numbers defined by the sequence c_i above. By theorem 1.11 the cost $A(\varphi)$ is larger or equal to the minimal cost.

Now, let n be an odd number and let T be the following sum:

$$T = a_1 a_n + a_2 a_n + a_3 a_{n-1} + \ldots\ldots + a_i a_{n-(i-2)} + \ldots\ldots + a_{\lceil \frac{n}{2} \rceil}\, a_{n-(\lceil \frac{n}{2} \rceil - 2)} + a_{\lceil \frac{n}{2} \rceil}\, a_{n-\lceil \frac{n}{2} \rceil} + a_{\lceil \frac{n}{2} \rceil + 1}\, a_{n-(\lceil \frac{n}{2}+1 \rceil}$$

$$+ \ldots\ldots\ldots + a_i a_{n-i} + \ldots\ldots + a_{n-2} a_2 + a_{n-1} a_1$$

Again, it is easily checked that the sum above corresponds to a Hamiltonian cycle. The first factors in this sum can be described by the following sequence:

$$d_i = \begin{cases} a_i & \text{for } i = 1,2,3,\ldots,\left\lceil \frac{n}{2} \right\rceil \\ a_{\left\lceil \frac{n}{2} \right\rceil} & \text{for } i = \left\lceil \frac{n}{2} \right\rceil + 1 \\ a_{i-1} & \text{for } i = \left\lceil \frac{n}{2} \right\rceil + 2,\ldots,n \end{cases}$$

These numbers form an ascending sequence. The second factors in the sum T form the following sequence:

$$e_i = \begin{cases} a_n & \text{for } i = 1 \\ a_{n+2-i} & \text{for } i = 2,3,\ldots,\left\lceil \frac{n}{2} \right\rceil \\ a_{n+1-i} & \text{for } i = \left\lceil \frac{n}{2} \right\rceil + 1,\ldots,n \end{cases}$$

As in the previous case, the sum T formed by multiplying the numbers from the two sequences pair ways, gives the smallest possible cost using these two sequences. In a similar fashion as in the even case, one can show that any other Hamiltonian cycle will lead to a cost where the same sequences are used, but where the elements in the sequence e_i are permuted in an non-descending order.

QED

Usually one wants to minimise the cost of the Hamiltonian cycles. In some cases however, finding the maximal cycles will be necessary, as it will be in some applications in chapter 3. A maximal cycle is usually found by multiplying each entry in the original cost matrix by minus one. Then one solves TSP in this new matrix. However, when we are dealing with polynomial cases of TSP, such a procedure may destroy the polynomial algorithm. This will be the case for the class of symmetric product matrices and the algorithm above. Multiplying each of the entries in the matrix by minus one, gives a matrix which is not a symmetric product matrix. Neither will it suffice to multiplying the entries of the underlying vector by minus one. The last procedure will give exactly the same matrix as the original one. Hence, in order to find the maximal Hamiltonian cycle in a symmetric product matrix, one must create a new procedure. This is done below.

Algorithm for finding maximal Hamiltonian cycles in symmetric product matrices.

To illustrate the algorithm, consider the cycle below. The cost of this cycle will be:

$$S = a_1a_2 + a_1a_3 + a_2a_4 + a_3a_5 + a_4a_6 + a_5a_7 + a_6a_8 + a_7a_9 + a_8a_{10} + a_9a_{10}$$

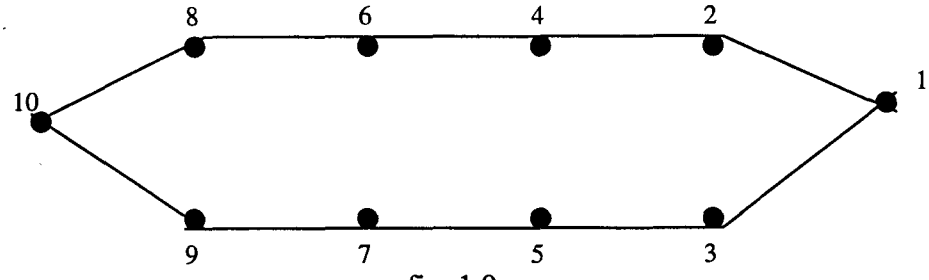


fig. 1.9

The first factors in the parts in S constitute a sequence of numbers in ascending order. The smallest co-ordinate in the underlying vector occurs twice and the largest is not present in the sequence. The second factors in the parts in S also constitute a sequence of numbers in ascending order. Here the smallest co-ordinate in the underlying vector is not present, whilst the largest co-ordinate occurs twice. By theorem 1.11, S constitutes the smallest possible sum of pair ways products ever obtainable from these two sequences. In general let n be any number of nodes in the graph. Consider the sum

$$T = a_1a_2 + a_1a_3 + a_2a_4 + \ldots\ldots + a_ia_{i+2} + \ldots\ldots\ldots + a_{n-3}a_{n-1} + a_{n-2}a_n + a_{n-1}a_n$$

It is easily checked that this sum is reflecting the cost of a Hamiltonian cycle in the graph. Moreover, the first factors in the parts in T, can be defined as

$$b_i = \begin{cases} a_1 & \text{for } i = 1 \\ a_{i-1} & \text{for } i = 2,3,\ldots,n \end{cases}$$

Note that this sequence is ascending and that the smallest of the co-ordinates in the underlying vector occurs twice and that the largest is not present. The second factor in the parts of T can be defined as

$$c_i = \begin{cases} a_{i+1} & \text{for } i = 1,2,\ldots,n-1 \\ a_n & \text{for } i = n \end{cases}$$

Note that this sequence is ascending too and that the smallest of the co-ordinates in the underlying vector is not present and that the largest is present twice. Hence, by theorem 1.11 the pair ways products of these two sequences yield the smallest sum.

Now, let ψ be any Hamiltonian cycle in the graph. Then the cost of this cycle will be:

$$A(\psi) = a_1a_{\psi(1)} + a_2a_{\psi(2)} + \ldots\ldots\ldots + a_na_{\psi(n)}$$

Just as in the minimising cases, this sum can be re-written such that the first factors in the different parts will be identical to the sequence b_i. The second factors will then be some permutation of the other sequence. Hence, the cost will be less according to theorem 1.11.

QED

For product matrices there is evidently a sharp border between the asymmetric and the symmetric cases, the former being NP-hard and the latter polynomial. Again it turns out that changing from a traditional sum TSP to a bottle-neck TSP the situation changes. The bottle-neck TSP for so called ordered asymmetric product matrices is polynomial with an algorithm restricted by $O(n)$, see van der Veen, 1993.

Constructing a product matrix given the vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ is of course very simple. On the other hand, if one has any matrix, one does not know á priori whether its entries can be described by vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ or not, that is we need some criterion to decide whether a given matrix is a product matrix or not. The following lemma gives such a criterion.

Lemma 1.3

Let C be a matrix, R_i and K_j the sum of the entries on row no. i and column no. j, and R the sum of the rows. Then

$$C \text{ is a product matrix iff } c_{ij} = \frac{R_i K_j}{R}$$

Proof: Suppose $c_{ij} = \frac{R_i K_j}{R}$. Define $a_i = \frac{R_i}{\sqrt{R}}$ and $b_j = \frac{K_j}{\sqrt{R}}$. Then clearly the vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ define the matrix C. Suppose that $C = \lfloor a_i b_j \rfloor_{nxn}$ for some vectors $\boldsymbol{a}$ and $\boldsymbol{b}$. Then, $R_i = a_i \sum_{j=1}^{n} b_j \;\; K_j = b_j \sum_{i=1}^{n} a_i$ and $R = \left(\sum_{i=1}^{n} a_i \right) \left(\sum_{j=1}^{n} b_j \right)$. Hence, $a_i = \frac{R_i \sum b_i}{R}$ and $b_j = \frac{K_j \sum a_j}{R}$. The result follows.

QED

Corollary 1.4

Any symmetric product matrix is doubly graded and its bottle-neck TSP is polynomial solvable.

Proof:

By renumbering the elements of the vector $\boldsymbol{a}$ such that they form a decreasing sequence. Then clearly the resulting product matrix will be doubly graded and then by theorem 1.6, the bottle-neck TSP is polynomial solvable.

QED

There seems to be a sharp border between asymmetric and symmetric product matrices. However, there exists a subset of the asymmetric class – which in general is NP-hard – that is polynomial solvable, namely the so-called ordered product matrices.

Hence, let $\boldsymbol{a}$ and $\boldsymbol{b}$ be two vectors with n elements and define the elements of a product matrix by

$$c_{ij} = a_i b_j$$

A matrix C is said to be an ordered product matrix iff

$$0 \leq a_1 \leq a_2 ..... \leq a_n \text{ and } b_1 \geq b_2 \geq ..... \geq b_n \geq 0$$

This class of TSP is a subset of the so-called Monge matrices, see sub-section 1.7.7.

1.7.6 Convex-hull- and -line TSP

In sub-section 1.6.2.1 the so-called convex hull heuristic was briefly touched upon. Given an Euclidean metric it is easy to construct the convex hull, and if all nodes take part in the convex hull, one has found the optimal solution. However, if there are nodes in the interior of the hull, one does not know in general how to insert these nodes between the nodes on the hull in order to obtain an optimal solution. Deineko et alt., 1994 have solved a special case of this situation, where n - m nodes lie on the boundary of the convex hull and the other m nodes on a line segment inside the hull. It turns out that there exists an $O(mn)$ time algorithm for this special case. A variant of the convex hull and a line-segment inside the hull is a situation where the nodes lie on a small number of N lines in the Euclidean plane. Rote, 1992, has shown that there exists a dynamic programming approach in time n^N that solves this problem.

1.7.7 Pyramidal Tours

A cyclic permutation is called a pyramidal cycle if it is of the following kind:

$$1 - i_1 - i_2 - ..... - i_r - n - j_1 - j_2 - ..... - j_{n-r-2} - 1$$

where

$$i_1 < i_2 < ..... < i_r \text{ and } j_1 > j_2 > ..... > j_{n-r-2}$$

In other words, the salesman starts in node 1, and visits a certain number of nodes in increasing order according to the given numbering, reaching node n and then returning to node 1 visiting the remaining nodes in a decreasing order. The set of pyramidal cycles constitutes an exponential subset of all the Hamiltonian cycles among the n nodes. It turns out that to find the best of all such pyramidal cycles can be done in polynomial time.

Theorem 1.12

For any nxn matrix C, the problem of finding the shortest pyramidal cycle with respect to C is solvable in $O(n^2)$ time.

For a proof of the above theorem, see Lawler, 1985. Note that applying the theorem on any distance matrix – or rather applying the dynamic programming scheme behind the proof of the theorem - can be regarded as a heuristic approach to any instance of TSP.

Moreover, it turns out that it is possible to identify certain structures of cost matrices that guarantee the existence of a shortest cycle that is pyramidal. Hence, in cases where the cost matrix possesses such properties, the optimal solution can be found in polynomial time.

Several structures ensuring that there exists an optimal solution, which are pyramidal have been found during the last decades. These structures have been given different names by different authors, but we will adapt the names used by Burkard et alt.,1995.

The following structures will be reviewed:

- **Monge matrices**
- **Van der Veen matrices**
- **Demidenko matrices**
- **Supnic matrices**
- **Kalmanson matrices**
- **Generalised Distribution matrices**

Monge matrices.

A matrix C is called a Monge matrix iff

$$c_{ij} + c_{rs} \leq c_{is} + c_{rj} \quad \forall 1 \leq i < r \leq n \text{ and } 1 \leq j < s \leq n$$

Another name for such matrices is distribution matrices. The term 'Monge' was framed by Hoffman, who rediscovered an observation made by the French mathematician Gaspard Monge in 1781. Note that the class of so-called ordered product matrices mentioned above is a sub-class of the Monge matrices.

Van der Veen matrices

A symmetric matrix C is called a Van der Veen matrix iff

$$c_{ij} + c_{j+1,l} \leq c_{il} + c_{j,j+1} \quad \forall 1 \leq i < j < j+1 < l \leq n$$

Symmetric Demidenko matrices

A symmetric matrix C is called a symmetric Demidenko matrix iff

$$c_{ij} + c_{j+1,l} \leq c_{i,j+1} + c_{jl} \quad \forall 1 \leq i < j < j+1 < l \leq n$$

This class contains the class of symmetric Monge matrices as a sub-class. On the other hand, the class of Van der Veen matrices neither contains, nor is contained, in the class of symmetric Demidenko matrices.

Supnic matrices

A symmetric matrix C is called a Supnick matrix iff

$$c_{ij} + c_{j+1,l} \leq c_{i,j+1} + c_{jl} \leq c_{il} + c_{j,j+1} \quad \forall 1 \leq i < j < j+1 < l \leq n$$

The Supnick class is a sub-class of both the Van der Veer class and the symmetric Demidenko class.

The optimal cycle is always *1-3-5-.....-n-.....,-6-4-2-1*

Kalmanson matrices

A symmetric matrix C is called a Kalmanson matrix iff it full-fills the Kalmanson conditions below:

$$c_{ij} + c_{k,l} \leq c_{i,k} + c_{jl} \quad \forall 1 \leq i < j < k < l \leq n$$

$$c_{il} + c_{jk} \leq c_{i,k} + c_{jl} \quad \forall 1 \leq i < j < k < l \leq n$$

The first of these two conditions is exactly the symmetric Demidenko condition and so the Kalmanson class is a sub-class of this. The optimal solution is always given by 1-2-3-.....-,n-1,n,-1.

Deineko and Woeginger (1998) describe a solvable case for the quadratic assignment problem called the *Kalamanson-circulant QAP*. They use the Koopmanns-Beckmann formulation of QAP where the A matrix is restricted to a Kalmanson matrix and the matrix B is taken to be a symmetric circulant matrix generated by a decreasing function. Their subclass for the QAP specialise the original Kalmanson case above.

Generalised Distribution matrices

This class of matrices consists of four different types which all can be asymmetrical. All four types have the following condition in common:

(*) $$c_{ij} + c_{kp} + c_{pq} \leq c_{ip} + c_{pj} + c_{kq}$$

A generalised distribution matrix is of type I iff (*) holds for all nodes *i,j,k,p,q* in

$$S_I = \{i, j, k, p, q \in N | i, j, k < p < q; i \neq j; j \neq k\}$$

Similarly, a generalised distribution matrix is of type II, III and IV iff (*) holds for the nodes in

$$S_{II} = \{i, j, k, p, q \in N | i, j, q < p < k; i \neq j; j \neq q\}$$
$$S_{III} = \{i, j, k, p, q \in N | k < p < i, j, q; i \neq j; j \neq q\}$$
$$S_{IV} = \{i, j, k, p, q \in N | q < p < i, j, k; i \neq j; i \neq k\}$$

All the four types contain the class of Monge matrices as a sub-class.

Asymmetric Demidenko matrices

An asymmetric matrix C is called an asymmetric Demidenko matrix iff it satisfies the following four conditions for all nodes i,j,l with $i < j < j+1 < l$

$$(i) \quad c_{ij} + c_{j,j+1} + c_{j+1,l} \leq c_{i,j+1} + c_{j+1,j} + c_{jl}$$
$$(ii) \quad c_{ji} + c_{j+1,j} + c_{l,j+1} \leq c_{j+1,i} + c_{j,j+1} + c_{lj}$$
$$(iii) \quad c_{ij} + c_{l,j+1} \leq c_{i,j+1} + c_{lj}$$
$$(iv) \quad c_{ji} + c_{j+1,l} \leq c_{j+1,i} + c_{jl}$$

If C is symmetric, the four conditions above just reduce to one, namely the condition for the *symmetric* Demidenko class above.

Even if all the classes above are shown to have a polynomial algorithm for solving TSP, it is an NP-hard problem to test if a given instance of TSP is pyramidal or not. Baki and Kabadi (1999) describe a new class of pyramidal tours called *hereditary* and in addition give a polynomial algorithm to test whether a given instance of TSP is in the new class or not. Their paper also contains a good and updated overview for known classes of pyramidal tours. The same authors have in a different paper examined and identified some other classes of polynomial solvable classes of TSP, generalising the concept of permuted distribution matrices, see Kabadi and Baki, 1999.

1.7.8 Upper Triangular Matrices

A matrix C is called upper triangular iff $i \geq j \Rightarrow c_{ij} = 0$, that is all entries on and above the main diagonal are zero.

The TSP for such matrices is as easy as the assignment problem as shown by Lawler, 1971 in the following theorem:

Theorem 1.13

Let C be upper triangular and φ an assignment that is optimal to the constraint that $\varphi(n) = 1$. Then $C(\varphi)$ is equal to the length of an optimal cycle τ that can be easily constructed from φ.

Here again the construction of the optimal cycle is given as a part of the proof. Since finding the optimal assignment - even with the added constraint $\varphi(n) = 1$ - is polynomial, the class of upper triangular matrices is polynomial as well. The proof of the theorem can be found in Lawler et alt, 1985 as well as in the original paper.

Note that if we want to find an optimal Hamiltonian *path* in an upper triangular matrix, we can extend the matrix with an extra node and attach costs equal to zero from all nodes to this new one and vice versa. The new extended node will be upper triangular as well. Finding an optimal *maximal* cycle or path in upper triangular matrices can be done as easily as solving the minimising problem. Finally, to find optimal cycles or paths in *lower* triangular matrices can be done in the same way as for the upper triangular instances, simply by taking the transposed matrix as the starting point.

1.7.9 Brownian Matrices

A matrix C is called a Brownian matrix iff there exists two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ such that

$$c_{ij} = \begin{cases} a_i & \text{if } i < j \\ b_j & \text{otherwise} \end{cases}$$

Burkhard and Van der Veen, 1991, proved that this class of TSP is polynomial solvable.

Chapter 2

In this chapter, a relation between the cost matrix for an instance of TSP and the corresponding saving values is established. The relation is based upon the special polynomial TSP-case usually referred to as the constant-TSP, see sub-section 1.7.1, and the concept of savings as defined in sub-section 1.6.2. The relation is then utilised in different contexts.

2.1 The Saving Heuristic Revisited

As mentioned in section 1.6, the Clarke and Wright heuristic or the saving heuristic is usually applied on VRP, but it can also be applied to TSP. Whether used in the first or the second case, one is trying to maximise the sum of the so-called savings, hoping that the result of joining the edges will yield a low cost when measured in the original cost matrix. The values of the savings are not regarded important in themselves, but are only used to decide the sequence of the edges in producing a feasible solution to the problem. The original algorithm is not even concerned about finding a maximal combination of savings, but is a simple greedy heuristic with a metric different from the original cost matrix.

The values of these savings are based on the position of the depot and a customer's relative position to the depot and to each other. Applied to TSP we may choose any node in the graph as our depot.

Let d be any node in the graph. The savings relative to d are then given by (2.1),

(2.1) $$s_{ij}^{d} = c_{id} + c_{dj} - c_{ij}$$

In the original cost matrix C it will be assumed that $c_{ii} = 0$ for all i. It is easily checked that in the resulting matrix $S^d = \left[s_{ij}^{d}\right]_{nxn}$ calculated by (2.1), row no. d and column no. d are zero. Further, the savings can take any values, but will remain non-negative as long as the triangle inequality holds for the cost matrix C and C is non-negative.

(2.1) shows that any saving matrix is a linear admissible transformation of the original cost matrix. This means that the set of optimal tours with C as the cost matrix, will be the same as the set of optimal tours for any saving matrix. Further, the total ordering of all Hamiltonian cycles in the graph will be reversed when calculated in any of the

saving matrices compared to the original cost matrix. The last comments were already made by Berenguer, 1979, and Lenstra and Rinnooy Kan, 1979. This observation could lead to the conclusion that nothing much is obtained by calculating the savings.

Many variations of the saving heuristic have been proposed during the last decades. Due to the way the saving values are calculated and the greedy fashion of the heuristic, the procedure tends to match nodes close to each other and far away from the depot early in the process. Nodes situated close to the depot but relatively far from each other, tend to be connected late in the procedure. One has tried to counteract this effect by adding some kind of modification to the saving values. One such approach has been proposed by Paessens, 1988. He suggests the following modifications (in the symmetric case)

(2.1b) $$s_{ij}^{d} = (c_{id} + c_{dj}) - \gamma c_{ij} + \delta \left| c_{id} + c_{dj} \right|$$

where γ and δ are parameters to be chosen freely inside the intervals <0,3] and [0,1] respectively. Paessens reports better computational results. On the other hand this approach calls for substantially more experimentation and computer time, and one has of course no guarantee of finding the optimal solution. Therefore one choice of the parameters in a specific instance of TSP or VRP cannot necessarily be used in another instance.

Setting $\delta = 0$ and rewriting (2.1b) to get the asymmetric case, we get

(2.1c) $$s_{ij}^{d} = (c_{id} + c_{dj}) - \gamma c_{ij} \qquad \forall i, j$$

This kind of modification is used by Golden, Magnanti and Nguyen, 1977, and they call the parameter γ for the *route shape* parameter. The route shape parameter was introduced by Yellow, 1970. Again, the transformation (2.1c) is a linear admissible transformation. Gaskell, 1978, chooses $\gamma = 2$ and calls the transformation π_{ij}, and restricts the discussion to the symmetric case. The effect of the parameter γ is to modify the saving values in such a away that, if two nodes are far from the depot and relatively far from each other, the original saving values - at least when applied in VRP - still can choose to link the two nodes. By increasing the route shape parameter from zero, the modified saving values will decrease, or if one likes, greater emphasis is placed on the distance between the two nodes rather than their relative positions to the depot. The linear admissibility shows, however, that nothing much will be achieved by this. The ordering of the cycles will be the same, only reversed. But as observed by experiments, different cycles will appear applying the basic heuristic for different versions of the saving values.

(2.1b) is formulated for the symmetric case, and due to the absolute value sign in the last part, this transformation is not necessarily a linear admissible transformation. It is more difficult to find a motivation for this modification. Paessens gives no motivation apart from the fact that he gets better result for the instances treated. However, rewriting (2.1b) to treat the asymmetric case and replacing the absolute sign by an ordinary bracelet, gives

(2.1d) $$s_{ij}^{d} = (1+\delta)c_{id} + (1-\delta)c_{dj} - \gamma c_{ij} \qquad \forall i,j$$

showing that we again have a linear admissible transformation. In principle there should be no reason to restrict the value of the parameter δ only to the interval [0,1], but all values are in principle equally good. If $\delta > 0$, the effect in (2.1d) will be to increase the distance from node i to the depot and to decrease the distance from node j to the depot.

It has also been observed, noteably by Gaskell 1978, that savings with the same values, place the nodes in question on a hyberbola. This is easily seen by taking any of the saving versions and let the left-hand side be a constant, for example in (2.1d)

$$(1+\delta)c_{id} + (1-\delta)c_{dj} - \gamma c_{ij} = cons$$

As already mentioned in Ch.1, the values of the savings are only used to make a sequence of nodes and arcs. Traditionally, the sum of the used savings does not enter the calculations. The cost of the Hamiltonian cycle is calculated in the original cost matrix, and there is no mention in the literature whether a low cost in the original cost matrix always corresponds to a large cost of savings or not, until the results of Berenguer in 1979.

In this sub-section, we show that there exists a very precise relationship between such costs and the saving values. We start with the saving values as given by (2.1)

Let $$K^{d} = [k_{ij}^{d}] \text{ where } k_{ij}^{d} = c_{id} + c_{dj}$$

This gives the following straightforward relation between the three defined matrices.

(2.2) $$C + S^{d} = K^{d}$$

for $d=1,2,....,n$, disregarding the elements on the diagonal, which do not concern us.

Let φ be any cyclic permutation of the indices i and j, reflecting a Hamiltonian cycle in the graph. This permutation gives the cost of the same Hamiltonian cycle in the three different matrices C, S^{d}, and K^{d} or if one wants, in three different metrics.

The elements of the matrix K^{d} have the special structure of a constant-TSP. Hence every Hamiltonian cycle has the same cost. It is easy to see that the cost of any tour must be $K_d + R_d$, where K_d and R_d make the d-th column and row sum respectively of the cost matrix C. By the argument above, we then have the following theorem:

Theorem 2.1

For any permutation reflecting a Hamiltonian cycle in the graph

(2.3) $$C(\varphi) + S^d(\varphi) = K_d + R_d \quad \forall d$$

This simple result seems to have been overlooked in the literature. The equation (2.3) simply means that the cost of a Hamiltonian cycle, plus the cost of the corresponding cycle measured in any saving matrix, is a constant for every Hamiltonian cycle. Hence, the original cost matrix and any of its saving matrices can be seen as complementary.

Equation (2.3) shows that finding the maximal sum of savings reflecting a Hamiltonian cycle, will be as good as finding a minimal sum of the original cost elements. This can be taken as an alternative proof - and a much simpler proof - of Berenguer's result on linear admissible forms restricted to the saving heuristic. If φ^* is an optimal permutation, then $C(\varphi^*)$ will be the minimal cost and $S^d(\varphi^*)$ will reach its maximal value.

Hence, there seems to be a good motivation not only for calculating the savings, but one can use any saving matrix as an alternative to the cost matrix C, that is, we can equally well try to find the maximal sum of savings constituting a Hamiltonian cycle.

The property of equation (2.3) can also be expressed in the following way:

The radius vectors of the cost in C, and in any saving matrix for any Hamiltonian cycle, will meet on the periphery of an ellipse. The situation is illustrated in figure 2.1 below.

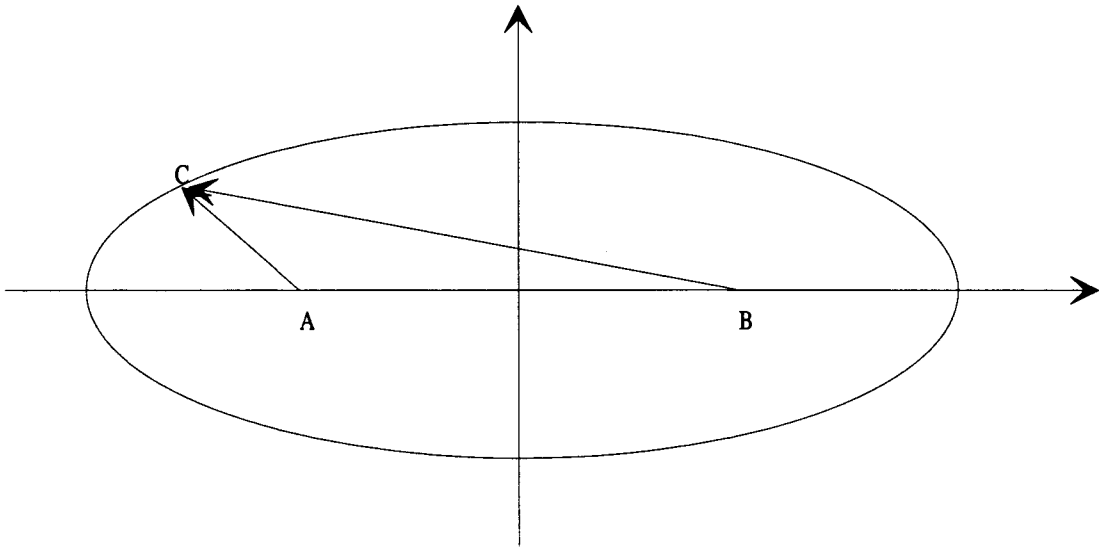


fig. 2.1

In the figure, for example, AC will correspond to the cost of a given tour in C and BC to the same tour in a saving matrix. Taken together $AC + BC = K_d + R_d \quad \forall d$, where the right hand side characterises the ellipse. Of course, not every point on the periphery of the ellipse will correspond to a Hamiltonian cycle, since the number of Hamiltonian cycles is finite. Neither should one expect the distribution of cycles along the periphery to be uniform in any sense.

Anyway, it is a bit surprising to observe that TSP - being a very typical discrete problem - has such a simple and close relation with a very old part of mathematics such as conic sections.

(2.3) holds for all instances of TSP. If C is restricted to symmetric matrices we have

Corollary 2.1

If the cost matrix C is symmetrical

(2.4) $$C(\varphi) + S^d(\varphi) = 2R_d \quad \forall d$$

Proof: In symmetric matrices every row sum is equal to the corresponding column sum.

QED

A different version of (2.3) can be obtained from (2.1d) in exactly the same way giving

(2.5) $$\gamma C(\varphi) + S^d(\varphi) = (1+\delta)K_d + (1-\delta)R_d \quad \forall d$$

which gives a slightly different shaped ellipse than displayed than (2.3).

2.1.1 Weighted Saving Matrices

In (2.5) the matrix C is modified by the shape parameter γ, and the right hand side is modified by the parameter δ. The savings as such are not modified, but are just consequences of the other modifications. Alternatively, one could modify savings directly as C is modified by the shape parameter.

Now, let $\boldsymbol{w}$ be a vector with n elements, where all the elements are non-negative and equal to or less than one, such that the sum of the elements w_i is equal to one. Multiplying (2.2) by w_d, taking the sum over $d = 1,2, \ldots, n,$ and applying the result to any Hamiltonian cycle, we get:

(2.6) $$C(\varphi) + \sum_{d=1}^{n} w_d S^d(\varphi) = \sum_{d=1}^{n} w_d (K_d + R_d)$$

Equation (2.6) states that we have unlimited possibilities for choosing the saving matrix, since the second part of the left hand side can be interpreted as a weighted matrix of the n different saving matrices. Alternatively, one can say that choosing a specific weight vector $\boldsymbol{w}$, defines a new depot different from any of the original nodes.

In a similar way we can of course use (2.2) to get

$$C+\sum_{d=1}^{n} w_d S^d = \sum_{d=1}^{n} w_d K^d = K \tag{2.7}$$

for the matrices involved, again disregarding the diagonal elements. Note also that the shape parameter can be replaced by a weighted saving matrix with $w' = \alpha w$. Further, equation (2.6) incorporates all previous attempts to modify the saving values.

Now. Let i and j be two indices. Then, by (2.2) we get - apart from the diagonal -

$$S^i - S^j = K^i - K^j \tag{2.8a}$$

or if one prefers to use (2.3)

$$S^i(\varphi) - S^j(\varphi) = (K_i - K_j) + (R_i - R_j) \tag{2.8b}$$

Equation (2.8b) states that the difference between two sets of savings corresponding to the same permutation equals the sum of the differences between the corresponding column and row sums. If C is a symmetric matrix we get:

$$S^i(\varphi) - S^j(\varphi) = 2(R_i - R_j) \tag{2.8c}$$

Further, let φ and ν be two permutations. Since the right hand side of (2.3) or (2.6) is the same for every permutation, we have:

$$C(\varphi) - C(\nu) = S^d(\nu) - S^d(\varphi) \tag{2.9}$$

Equation (2.9) states that the difference between two Hamiltonian cycles measured in the original cost matrix C, equals the difference between the corresponding savings in any of the saving matrices, but with an opposite sign. Equation (2.9) reflects the fact that the savings are linear admissible transformations of the original cost matrix.

2.2 Direct Applications

Equations (2.3) or (2.6) open up for several potential applications, apart from the obvious observation that the sum of the costs of a Hamiltonian cycle in C, and the corresponding savings in any of the saving matrices, is a constant. The cost directly incurred by the original cost matrix is complementary to the cost incurred by any of the infinitely many saving matrices. In principle, one can equivalently use any

(weighted) saving matrix in attacking the TSP problem, and hope for easier or faster roads to an optimal solution. What has been achieved by these transformations, is that the original problem has been transformed into a new metric that may reveal other features of the problem than the original cost matrix could do alone. Some of these possibilities are discussed in the following sub-sections.

2.2.1 Solving the TSP with Saving Matrices

As mentioned above, one can instead of minimising the costs of the original cost matrix, maximise the savings in any of the saving matrices or a weighted combination of these, in a model for TSP. This shows that the basic idea of Clarke and Wright was a sound one, but was not fully utilised in the beginning. However, the results obtained by trying this can be very limited.

Let F be any set of facets added to the restrictions in model 1 in chapter 1. This will give some solution, possibly a fractional, creating a lower bound for the TSP when the original cost matrix C is used in the objective function. Let $\boldsymbol{U}$ be the solution vector to this, and $\boldsymbol{w}$ a weight vector for the savings.

The elements of the weighted matrix K in (2.6) are given by:

$$k_{ij} = \sum_{d=1}^{n} w_d (c_{id} + c_{dj}) \tag{2.10}$$

It is easily seen that as long as (1.2) and (1.3) holds, then the solution vector $\boldsymbol{U}$ applied on this matrix, gives the same value as the right hand side of (2.6). This means that for a given set of facets, the value of the solution arising from the original matrix C, plus the value of the solution from any weighted saving matrix, always adds up to the right hand side of (2.7). In other words: It does not matter which cost matrix we use, the bound created with F will basically be the same.

Then, in order to utilise some (weighted) saving matrix, one must use relaxations for the TSP not containing (1.2) or (1.3) or their equivalents. Such formulations are usually weaker than those above, in the sense that the bound given by a set of facets is poorer.

We have tried with a 1-tree relaxation on a small example. This ended up with more than forty sub-tour eliminating facets when we used the original cost matrix. Using one of the saving matrices, we had to use approximately the same number of facets of the same kind, but some of the facets were different in the two cases. Nothing much seems to be gained from such a procedure.

2.2.2 Applications to Heuristics

The main result so far is that any saving matrix of the original matrix C, can be viewed as a complementary matrix, that is we can apply the saving matrix instead of the original matrix. A good solution - that is a solution with a high saving cost - will unambiguously correspond to a low cost in C.

Hart and Shogan, 1987, argue that the use of several heuristics is to be preferred when trying to solve combinatorial optimisation problems, since we then are able to generate several sub-optimal solutions and it is also preferable to have several mathematical functions to govern the heuristic. It will be shown below, that transformation to savings - weighted or not - gives a huge amount of new possibilities for heuristics for TSP. Since any saving matrix is as good as the original cost matrix, many of the elementary heuristics for TSP applied on the cost matrix C can now be applied to the saving matrices with minor changes in the procedure. As a consequence, quite different Hamiltonian cycles can occur, sometimes better and sometimes worse. Moreover, any application of such heuristics can be applied an unlimited number of times, since we can choose an infinite number of vectors $\boldsymbol{w}$. Choosing different weighted saving matrices may give very different cycles.

2.2.2.1 Nearest Neighbour

Instead of taking the nearest node in C, we take the most expensive node in the chosen saving matrix. This particular heuristic can only be applied n times on the original cost matrix, but $n(n - 1)$ times on the saving matrices, even if we restrict ourselves to choosing one of the weights to be one and all others equal to zero. Moreover, it can be applied an infinite number of times if we choose a weighted saving matrix of some kind.

2.2.2.2 Insertion Heuristics

All such heuristics involve choosing a new node that is not a part of an existing sub-tour, using some criterion. Then, using some other criterion, this new node is made a part of a new sub-tour containing all the nodes from the old sub-tour, plus the new one.

Let E be the set of edges in the current sub-tour and k the node "closest" to this sub-tour. Then one shall insert the node k between two nodes in the existing sub-tour. For instance, this can be done by calculating the cost increases defined by:

$$\Delta_C(i,k,j) = c_{ik} + c_{kj} - c_{ij} \tag{2.11}$$

for all edges (i, j) in E, and then choosing the minimal increase as the criterion for insertion.

Translated to a saving matrix we get:

$$\Delta_{S^d}(i,k,j) = s^d_{ik} + s^d_{kj} - s^d_{ij} = 2c_{kd} - (c_{ik} + c_{kj} - c_{ij}) = 2c_{kd} - \Delta_C(i,k,j) \tag{2.12}$$

where one should choose the edge (i, j) that gives the highest value, that is the "best increase" in the savings. (2.12) shows that the best (minimal) triple also should be used in the saving variant of the heuristic, since this choice gives the best (maximal) triple. This means that we may get identical cycles in both cases. This will happen if, from a given sub-tour, one must choose the same new node k.

However, this will not always be the case. Since the choice of the new node is done in a saving matrix, and this node can be different from the one that should be chosen in the original matrix, as is shown by the examples in chapter 5. Hence, as soon as a selection criterion in the original cost matrix is modified to be applicable on a saving matrix, the result can be very different, even if the insertion criterion will give the same result for identical nodes.

This means that the following insertion heuristics mentioned in chapter 1 can behave differently when applied to saving matrices:

- Nearest insertion
- Farthest insertion 1
- Farthest insertion 2
- Farthest insertion 3
- Cheapest insertion
- Smallest sum insertion
- Largest sum insertion

On the other hand, there is no point in applying "random insertion" to a saving matrix using insertion criterion (2.12), since the selection criterion in this case does not involve any cost evaluations.

Again, in their original versions, all these heuristics can only be applied n times using the original cost matrix, but using savings matrices one has the possibility of applying it $n(n-1)$ times, or even an infinite number of times.

2.2.2.3 Convex Hull

This heuristic is based on an Euclidean cost matrix that makes it possible to find a convex hull. Changing the cost matrix to any saving matrix does not ensure such a possibility, since not even the triangle inequality necessarily holds in a saving matrix. Hence, the starting point of this heuristic – that is, finding the convex hull - must be done in the original cost matrix C. Whether different cycles can be obtained by using some selection criterion in a saving matrix, compared with a similar selection criterion in the original cost matrix, will then depend on the selection criterion only.

2.2.2.4 The Saving Heuristic

In the original saving heuristic the savings are, as previously mentioned, ordered in a decreasing sequence of lower and lower values. The savings are chosen such that in the end one gets a path containing all the nodes, apart from the chosen depot node.

We are now in principle able to apply this heuristic an infinite number of times using a weighted saving matrix as input.

In the original application, it is not necessary to calculate the savings connected with the depot, since those savings are zero, and what we are searching for is a spanning

path, signifying that, we have to find *(n-1)* edges. By using a weighted saving matrix we are in a way creating a dummy node outside the underlying graph, and we must search for a full Hamiltonian cycle, that is we have to find *n* different edges. None of the weighted saving values need to be zero. The effect of the shape parameter in (2.5) is the same, creating a depot outside the underlying graph.

Remark:

If one accepts the basic idea of the original procedure in the saving heuristic, one may as well, with equation (2.3) in mind, do the same thing in the original cost matrix. The elements of *C* are ordered in an increasing sequence. One then chooses elements with larger and larger values until one in the end gets a cycle. Of course, edges creating sub-tours and nodes with in- or out-degrees larger than two have to be avoided. Such a heuristic will be a version of the nearest neighbour heuristic and viewed at the background of (2.3) to be as good or bad as any application of the original saving heuristic. This kind of heuristic is not directly mentioned in any of the references treating heuristics for TSP. In principle, there will be no difference between such a heuristic and the Clarke and Wright heuristic.

2.2.2.5 The Loss Heuristic

In the loss heuristic, the so called loss function *LOSS(j)* is basically calculated as the difference between the nearest and the next nearest node, to any node not already incorporated in the interior of a path. We can now do the same based on any saving matrix, ie, calculate the difference between the savings corresponding to the node with the largest saving, minus the node with the next largest saving. and then incorporate the node with smallest loss in savings. For instance, based on fig 1.4 and a node from category B we get:

$$LOSS^s(j) = s^d_{jt} - s^d_{js} = c_{dt} - c_{ds} - (c_{jt} - c_{js}) \quad \forall j,s,t \neq d \tag{2.13}$$

The criterion above clearly shows that the chosen nodes can be quite different from the chosen nodes using the original cost matrix, and can be different from one saving matrix to another. Hence, using saving matrices opens up for many more applications with the same kind of thinking.

2.2.2.6 Heuristics Based on Minimal Spanning Trees

Kim's and Christofides' heuristics

There are two such heuristics. Kim's heuristic is based on a doubling of the MIST for C, and then creating an Eulerian cycle in this network. Following the Eulerian cycle one creates a Hamiltonian cycle by taking shortcuts whenever necessary in order to avoid nodes already visited. The second one - the so-called heuristics of Christofides - is also based on the MIST, but instead of doubling the tree, one creates an Eulerian graph by making a minimal matching among the odd nodes in the MIST.

In the same way, one can use any of the saving matrices by constructing MAST instead of the MIST, and then apply the two heuristics to this tree, making a maximal matching in the Christofides' case. Again, the possibilities for applying the heuristics can be multiplied by n, since the MASTs in different saving matrices can be quite different from the MIST in the original matrix and different from each other.

2.2.2.7 Stretching Tree Heuristic - A New Heuristic Based on MIST

This heuristic has some resemblance with the two above since it tries to utilise the structure of the MIST by trying to keep as many of the edges from MIST as possible, but when this is not possible, other edges in the graph are used. The differences are that one does not start with an Euler tour and that only a Hamiltonian path is created. When this is accomplished, the path is closed by matching its end points.

Step 1: Construct the MIST

Step 2: Identify the leaves of the tree. Choose one of them, say i_1. Let P be a subset of the nodes. Put $P = \{i_1\}$

Note that there is always at least two leaves in a tree, so step 2 can always be performed.

Step 3. Follow the tree from the chosen node until a node with a degree larger, or equal to three, is encountered. Call this node j. Such nodes are called *junctions*. Update P so it contains all nodes passed so far, so that P contains all the nodes between the starting leave and the first found junction.

Step 4 : Identify all nodes $A = \{i_1, i_2, i_3 \ldots\ldots i_k\}$ not in P, and adjacent to the junction j. Note that k is larger or equal to two. Find the pair (s,t) such that this pair corresponds to

$$\min_{k,l \in A}\{c_{kl} - c_{kj}\}$$

Delete the edge (s,j) from the tree and add the edge (s,t). The degree of the junction j is by this reduced with one.

Step 5: If the degree of j is equal or larger than three, go to step 4. If the degree of j is two, go to step 6.

Step 6 : Add j to the set P. If $P = N$, then stop. If not, follow the tree until a new junction is found and P is different from N. Then go to step 4. If $P = N$, go to step 7.

Step 7: Close the Hamiltonian path, by connecting the ending nodes of the path. Stop

Note, that since the number of degrees in any tree is $2(n-1)$ and that the "stretching" part in step 4 involves only a polynomial number of calculations and possibilities, the algorithm will stop in polynomial time.

The stretching heuristic based on saving matrices.

The stretching heuristic can of course be applied to any saving matrix starting with MAST instead of MIST, and maximising instead of minimising in step 4. Note that as long as the saving matrix is based on one of the nodes in the graph and not on a weighted saving matrix, the saving values corresponding to row and column d are zero. Hence, the construction of the Hamiltonian path can be done in the graph without including the chosen depot node and the closing of the created Hamiltonian path will be done without adding extra savings.

2.2.2.8 Tour Improvement Heuristics

Any tour improvement procedure that can be applied to C, can of course also be applied to the saving matrices by interpreting "better" as "more expensive". However, as will be shown below, the most used tour improvement heuristic - the so called k-change - will give the same result, whether it be applied to a Hamiltonian cycle with costs in C, or to the correspondent cycle based on a saving matrix.

The reason for this is that a *k-change* procedure does not involve any selection or insertion criteria based on costs. Starting with a certain Hamiltonian cycle, k edges or arcs are deleted and replaced by k new edges or arcs, such that a different Hamiltonian cycle comes forth. Many different cycles can be constructed in this way if k is larger than two, but the construction does not involve any evaluation of costs. So, we have the same set of new cycles independently of the underlying cost matrix, this being the original one or any of the saving matrices. After every new cycle has been constructed, one calculates the cost. Whether one does this in the original cost matrix, or in any of the saving matrices, does not matter according to (2.3).

A variant of the *k-change* procedure

The standard procedure above depends only on the parameter k and the starting cycle, constructing all possible new Hamiltionian cycles, and then choosing the best one as the result of the heuristic, and can be modified in different ways. One of these is to calculate the cost of each new cycle at once, and as soon as one finds one with a smaller cost, this new cycle is used as an input cycle, starting the procedure from the beginning again. But this will not have any consequences as far as the different cost matrices are concerned. The cost matrices considered here are equivalent and irrelevant as long as no selection or insertion criteria are used.

However, if the parameter k becomes large, the number of ways the k edges can be chosen becomes large, and the number of different cycles that can be constructed becomes huge, at least when the number of nodes is large or reasonably so. In order to avoid too many calculations one can have some selection criteria for which edges to delete from the input cycle. One such possibility is:

Selection criteria: Delete the k edges with the largest sum in C

Using a saving matrix, the criteria will be

Selection criteria: Delete the k edges apart from edges connected to the chosen depot, which give with the smallest sum

In this way we will obtain different starting points for the main procedure.

Final remark

In the previous sub-sections, different heuristics have been discussed in the context of alternative but equivalent cost matrices. Especially, it has been demonstrated that a large number of such cost matrices can be created, and by examples show that the different cost matrices create different cycles when applied in at least some of the treated heuristics.

An interesting open question to ask is the following: Given a heuristic, is it possible to find a specific cost matrix - say a weighted saving matrix - such that the optimal solution will be found by the heuristic?

If the answer to this question is "yes", then the optimal solution can be found in polynomial time. On the other hand, the procedure for finding such a matrix may be – and probably is - non-polynomial. If the answer to the question is "no", there is still the possibility that subsets of instances of TSP have such a property. However, this open question will not be pursued in this thesis.

2.3 Savings and other Sub-graphs besides Cycles

We have seen above, that there is a very precise relationship between the cost of any Hamiltonian cycle measured in the original cost matrix, and any saving matrix. A natural question to ask, is whether there exists similar relationships if the sub-graph is not a cycle, but some other well defined structure. In the following sub-sections some of these questions are answered.

2.3.1 Savings and Assignments

An assignment is characterised by requiring one arc out from each node and one arc into each node, but one does allow sub-cycles as opposed to a Hamiltonian cycle. However, the cost of any assignment in a constant-TSP matrix will have exactly the same value as any of the Hamiltonian cycles. Hence the equations in (2.3) still hold for assignments. This means that (2.3) can be applied in the VRP case, see section 2.7. Further, if a certain assignment is a lower bound for a TSP in the original cost matrix, the same assignment will be an upper bound for the TSP measured in any saving matrix. We can calculate the value of any of them with the help of the other and (2.3)

2.3.2 Savings and Spanning Paths

Specifying the sub-graph to be a Hamiltonian path, where the starting and terminating node is known, we can still find a relation similar to (2.3)

Let $\varphi_p(s,t)$ denote a Hamiltonian path starting in node s and terminating in node t. We then have the following theorem:

Theorem 2.2

For any Hamiltonian path $\varphi_p(s,t)$ in a complete graph

$$C(\varphi_p(s,t)) + S^d(\varphi_p(s,t)) = K_d + R_d - (c_{td} + c_{ds}) \quad \forall d \tag{2.14}$$

Proof: Let φ be the Hamiltonian cycle obtained by adding the arc t-s in the given path. The cost of this cycle in the matrix K is then $K_d + R_d$. The cost of the arc t-s in K is $c_{td} + c_{ds}$.

QED

The theorem shows that finding the optimal Hamiltonian path when the starting and terminating nodes are given, can be done either by minimising the problem in the original cost matrix, or equally well in any saving matrix by maximising the sum of the savings. So, we have exactly the same situation as in the TSP case. Note also that the cost for any Hamiltonian path $\varphi_p(s,t)$ is the sum of the path measured in the original cost matrix plus the sum of the corresponding saving values for any of the savings matrices, and will be on the periphery on an ellipse as described in figure 2.1. The only difference is that for the same cost matrix C, the ellipse will be somewhat smaller due to the deduction $c_{td} + c_{ds}$ in (2.14) compared to (2.3).

Let $\varphi_p(s,-)$ and $\varphi_p(-,t)$ denote Hamiltonian paths where only the starting node or the terminating nodes are specified, respectively and let φ_p denote a Hamiltonian path where neither the starting nor the terminating nodes are specified. If one of these cases occurs, relation (2.14) must be modified and it is not possible to obtain equations for the relationship between the costs measured in the original cost matrix and the saving matrices. But we can still obtain upper and lower bounds for such problems. These bounds are given in corollary 2.2.

Corollary 2.2

For any Hamiltonian path of the type $\varphi_p(s,-), \varphi_p(-,t)$ and φ_p, we have for all d

$$K_d + R_d - (\max_{j \in \{s,d\}} \{c_{jd}\} + c_{ds}) \leq C(\varphi_p(s,-)) + S^d(\varphi_p(s,-)) \leq K_d + R_d - (\min_{j \in \{s,d\}} \{c_{jd}\} + c_{ds})$$

$$K_d + R_d - (\max_{j \in \{t,d\}} \{c_{dj}\} + c_{td}) \leq C(\varphi_p(-,t)) + S^d(\varphi_p(-,t)) \leq K_d + R_d - (\min_{j \in \{t,d\}} \{c_{jd}\} + c_{td})$$

$$K_d + R_d - \max_{i,j \neq d} \{c_{id} + c_{dj}\} \leq C(\varphi_p) + S^d(\varphi_p) \leq K_d + R_d - \min_{i,j \neq d} \{c_{id} + c_{dj}\}$$

Proof: The relations follow immediately from the proof of theorem 2.2.

QED

Observe that we also can find Hamiltonian paths as described in section 1.7, by extending the graph by some artificial node and adding arcs or edges with special values.

Suppose that one knows the starting and terminating nodes, and chooses to assign a sufficiently large negative value $-M$ to the arc (t,s). In this case, at least one of the saving values will change for every choice of the depot and if the depot is chosen to be either s or t, the right hand side of (2.3) will change as well. The second option is to add a new node and arcs attaching this new node to the starting and terminating nodes. Since no arcs are added to the other nodes in the original graph, (2.3) is difficult to apply, since to calculate the savings and the right hand side of (2.3) one needs a complete graph.

The same will be the case when either the starting node or the terminating node is known. In this case, a new node is added and we add new arcs, but not in a way that creates a complete graph. However, if neither the starting nor the terminating node is known, then we add a new node and add arcs with zero costs to all nodes in the old graph. By this procedure, the right hand side of (2.3) will have the same value, but all the involved matrices will have the size $(n+1)x(n+1)$, and everyone of the saving matrices will have new elements $s^d_{i,n+1} = c_{id}$ and $s^d_{n+1,j} = c_{dj}$, which then reflects the property of corollary 2.2

2.3.3 Savings and Trees

A path is a special kind of tree, so if we relax the premises of theorem 2.2 and corollary 2.2, we cannot hope for more than the statements in these propositions. However, it is still possible to obtain upper and lower bounds of any tree in a similar way as above.

Let τ denote a tree in an undirected graph with n nodes. We then have the following theorem:

Theorem 2.3

For every tree τ in an undirected graph with k nodes with degree 1, the following relation holds for every d:

$$R_d + (n-k)\min_i\{c_{id}\} \le C(\tau) + S^d(\tau) \le R_d + (n-k)\max_i\{c_{id}\} \tag{2.15}$$

Proof: Let deg(*i*) denote the degree of node *i*. In a tree, every node has a degree of at least one. In the matrix $K = [c_{id} + c_{dj}]$ the cost of τ is $K(\tau) = R_d + \sum_{i=1}^{n}(\deg(i)-1)c_{id}$. Since $C + S^d = K$ apart from the diagonal, we get (2.15).

QED

Note that in any tree the number of nodes with degree 1 is always two or more. A special case is a path, which has exactly two nodes with degree 1 and then (2.15) coincides with the last relation in corollary 2.2. Another special case is a hub. In such trees all nodes except one have degree 1, and (2.15) becomes

Corollary 2.3

For every hub in an undirected graph we have that

$$R_d + \min_i\{c_{id}\} \le C(\tau) + S^d(\tau) \le R_d + \max_i\{c_{id}\} \quad \forall d \tag{2.16}$$

2.4 Generalised Savings

In section 2.1 the basic relation (2.3) was established. These equations are based on manipulations with the cost elements of the original cost matrix *C*. The saving matrices as well as the constant-TSP matrix use elements only from this. However, we can do much of the same as has been done above, without relating the two constructed matrices so closely to the original one.

Let $K = [a_i + b_j]_{nxn}$ be a constant-TSP matrix where the *a*'s and the *b*'s are non-negative numbers such that $K > C$. Hence, we can construct a third matrix S^K, such that

$$s_{ij}^K = a_i + b_j - c_{ij} \tag{2.17}$$

This new matrix can be seen as a generalisation of an ordinary saving matrix, where we start from some kind of "virtual" node. The cost of travelling between this virtual node and node *i* is a_i if *i* is visited first, and b_j if *j* is visited as the second node. Note that (2.17) is a linear admissible transformation reversing the total order of the

Hamiltonian cycles, just as any of the traditional savings matrices do. The saving of taking both nodes on the same trip and in the sequence i before j, is then given by (2.17), which again leads to a similar relation as (2.3), namely

(2.18) $$C(\varphi) + S^K(\varphi) = A + B$$

where $A = \sum_{i=1}^{n} a_i$ and $B = \sum_{i=1}^{n} b_i$, respectively.

Since the a's and the b's can be chosen arbitrarily as long as $K > C$, we have created an infinite number of generalised saving matrices which can be used as starting points for all the heuristics mentioned in section 2.2. Note however, that in general we will not have zeros only in one of the columns and the corresponding row. If C is symmetrical and we want the generalised saving matrix to become symmetrical as well, we must let K be symmetrical. This will happen iff $a_i = b_i$ for all i.

2.5 Lower Bounds for TSP

In this sub-section we discuss some results based on section 2.1 where the aim is not to determine optimal solutions, but rather say something about the ramifications of the problem. Note, that if we know an upper bound, *UB,* for the optimal solution measured in a saving matrix, a lower bound *LB* for the optimal solution measured in the original cost matrix can be derived from (2.3) by

(2.19) $$C(\varphi^*) = K_d + R_d - S^d(\varphi^*) \geq K_d + R_d - UB(S^d) = LB(C)$$

2.5.1 Simple Lower Bounds for TSP

The simplest lower bound for TSP will be the smallest cost multiplied with the number of nodes. Let *LB0* denote this lower bound, that is

(2.20) $$LB0 = n \min_{i,j}\{c_{ij}\}$$

Applied on the original cost matrix this bound will usually be very poor. A similar upper bound on a saving matrix will be

(2.20b) $$UB0 = n \max_{i,j}\{s_{ij}^d\}$$

Hence, an alternative lower bound for TSP measured in the original cost matrix will by (2.19) be

(2.21) $$K_d + R_d - n \max_{i,j}\{s_{i,j}^d\} \leq C(\varphi^*)$$

Since the saving values in row and column no. d is zero, (2.21) can be sharpened to

(2.22) $$LB3 = K_d + R_d - (n-2)\max_{i,j}\{s_{i,j}^d\} \le C(\varphi^*)$$

It turns out that in concrete applications the lower bound given by (2.22) can be substantially better than (2.20)see chapter 5 for an example.

Another simple but better lower bound can be found by adding the smallest element in each row. Let *LB1* denote this bound, that is

(2.23) $$LB1 = \sum_{i=1}^{n} \min_j \{c_{ij}\}$$

and let *UB1* be the upper bound in some saving matrix defined by

(2.23b) $$UB1 = \sum_{i=1}^{n} \max_j \{s_{ij}^d\}$$

In a similar fashion as was done in (2.21), we can find an alternative lower bound

(2.24) $$LB4 = K_d + R_d - \sum_{i=1}^{n} \max_j \{s_{ij}^d\} \le C(\varphi^*)$$

Again (2.24) can be substantially better than (2.23), see chapter 5 for an example.

Finally, Webb, 1971, in his article about the so-called LOSS-heuristic, observes that for each node, we must add the value of two arcs. Then taking the sum of the two smallest costs in each line and dividing this by two and adding all these costs, we obtain a lower bound for TSP. This kind of thinking is the main motivation behind the LOSS heuristic. Let *LB2* denote this lower bound, ie

(2.25) $$LB2 = \frac{1}{2}\sum_{i=1}^{n} \min_{\substack{s,t \\ s \ne t}} \{c_{is} + c_{it}\}$$

Again we define a similar upper bound *UB2 by*

(2.25b) $$UB2 = \frac{1}{2}\sum_{i=1}^{n} \max_{\substack{s,t \\ s \ne t}} \{s_{is}^d + s_{it}^d\}$$

In a similar fashion as was done above we get another lower bound for TSP by combining (2.19) and (2.25) to get (2.26)

(2.26) $$LB5 = K_d + R_d - \frac{1}{2}\sum_{i=1}^{n} \max_{\substack{s,t \\ s \ne t}} \{s_{is}^d + s_{it}^d\} \le C(\varphi^*)$$

In order to illustrate the effects that can be obtained by the transformations in (2.22), (2.24), and (2.26) the following small example below will suffice.

Example 2.1

Let the cost matrix C be given by

	1	2	3	4	5
1	-	3	2	1	45
2		-	4	5	40
3			-	3	35
4				-	50
5					-

As can be seen from C the first four nodes are rather close to each other, and the fifth node is far from the other ones.

Node no.	*LB0**	*LB1**	*LB2**
Bounds with C**	5	42	47
1	66	70	81
2	65	69	72
3	76	77	80
4	64	61	77
5	58	-15	-

*) For the last five lines, (2.22), (2.24), and (2.26) are used [illegible] the bounds
**) In this line the three simple bounds are applied directly on the original cost matrix.

Apart from the last line in the table above, the lower bounds based on the saving matrices, give substantial better lower bounds than making these simple bounds on the original cost matrix only. Even the lower bound given by (2.22) seems to give very good results, but this is partly due to the low number of nodes, and the effect will probably be less marked when the number of nodes is increased.

Clearly the relationships between the lower bounds given by (2.20), (2.23) and (2.25) are $LB0 \leq LB1 \leq LB2$, and similarly, and hence $LB3 \leq LB4 \leq LB5$. However, to find general relationships between the first and second triple of the lower bounds seems to be difficult.

2.5.2 Lower Bounds for Symmetric TSP Based on Trees.

In this sub-section we will restrict ourselves to the symmetric case.

One such lower bound is the MIST on C, or even better, the minimal 1-tree.

A better lower bound for TSP can be obtained by deleting one of the nodes in the graph, constructing the MIST on the remaining nodes, and adding the two cheapest

edges. Doing this for all the nodes, one chooses the best lower bound. We will denote these 1-trees by 1-tree(*C(d)*) and they will be equal to

$$MIST(C(d)) + c_{d,i_1^*} + c_{d,i_2^*} . \tag{2.27}$$

where $c_{d,i_1^*} + c_{d,i_2^*}$ is the sum of the two cheapest edges attached to node *d* and *C(d)* is the cost matrix for the original graph minus node *d*. In general we have the following inequalities:

$$MIST(C) \leq 1-tree(C) \leq \max_d \{MIST(C(d)) + c_{d,i_1^*} + c_{d,i_2^*}\} .$$

Much in the same way one can use MAST + 1-tree on a matrix to obtain an upper bound for the maximal Hamiltonian tour, but the MAST alone will not always be an upper bound for this tour. However, with the help of the saving matrices we can prove the following lemma.

Lemma 2.1

Lower bounds and upper bounds for the minimal and maximal Hamiltonian cycle in any symmetric matrix C can be found by

$$2R_d - MAST(S^d) \text{ and } 2R_d - MIST(S^d)$$

respectively. Each d can be chosen independently of each other in the two expressions.

Proof: Take any saving matrix S^d. Delete the node *d* from the graph and construct MAST on the remaining graph. Then add the two most expensive edges connected to the node *d*. Since all the edges connected to *d* have costs zero, MAST(S^d) will be an upper bound for the maximal Hamiltonian cycle in S^d. By (2.3) we get

$$C(\varphi^*) = 2R_d - S^d(\varphi^*) \geq 2R_d - MAST(S^d)$$

The result for the lower bound follows. Similarly, we can make the minimal spanning tree on the saving matrix, and we get the upper bound for the maximal Hamiltonian cycle in *C*.

QED

One could try to utilise some weighted saving matrix in the same way. However, the result in lemma 2.1 is based on the fact that the costs of the edges connected to the depot are zeros. This will not be the case - in general - for any weighted saving matrix, and we then have to add the two most expensive edges to the MAST in the weighted saving matrix. This will in most cases lead to poorer lower bounds than the lemma 2.1 gives. The basic saving matrices transfer the information associated with the depot to the other nodes and leaves the depot independent of any of the other nodes as far as costs and savings are concerned.

Note that the same remark shows that solving the TSP with help of any non-weighted saving matrix, is equivalent to solving the maximum Hamiltonian **path** in the saving matrix.

Note also that the result in lemma 2.1 is a concrete application of an observation done by Held and Karp, 1969. They observe that adding real numbers a_i to the cost elements in a TSWP matrix, such that the edges have costs $c_{ij} + a_i + a_j$ does not alter the sequence of the costs of the Hamiltonian cycles, but may affect the identity of a minimum spanning 1-tree. In the case treated here the added costs is a part of the original cost matrix and is matched with finding a maximum spanning 1-tree in a closely related matrix, namely a saving matrix.

A natural question to ask in connection with the lower bounds in lemma 2.1 is how good these bounds are compared to other known lower bounds for TSP?

The following small example shows that in general the bounds given by lemma 2.1 for a given node *d* can be both better and worse than the bounds given by the more traditional graph considerations obtained by (2.27).

Example 2.2

Using the cost matrix of example 2.1, one finds that MIST(*C*) is 41 and *1-tree(C)* is 44. Making all lower bounds by (2.27) (second column) and the bounds found by the lemma, gives the following table:

node	$2R_d - MAST(S^d)$	$MIST(C(d)) + c_{d,i_1} + c_{d,i_2}$.
1	82	45
2	82	43
3	81	50
4	79	44
5	79	81

As can be seen from these results, the lower bounds given by the lemma are best for four of the five nodes. For node 5 however, (2.27) gives a better lower bound than the lemma. But it is still the case that the best lower bound is obtained by the lemma and not by (2.27). Comparing these results with the results obtained by the simple bounds in the previous sub-section, one will see that the latter ones compete very well with the bounds found by (2.27), and are only beaten by the best bound found by lemma 2.1.

It is my conjecture that this will always be the case. I have however, not been able to prove this. Neither have I been able to find a counter example. In chapter 5, the different lower bounds are compared using examples with more "normal" cost matrices.

Basically, the lower bound given by lemma 2.1 is corresponding to a minimal 1-tree in the original cost matrix. As mentioned above, a bound can be obtained from the original cost matrix when a node is deleted, a MIST is found by the remaining nodes, and the two smallest edges from the deleted node are added to the MIST. In our case here, however, the deleted node is the depot node. Since the savings from a depot node to any other node are zero, the two bounds coincide. However, given a saving matrix, one may of course delete any node from the graph, find the MAST among the remaining nodes, and then add the two edges with the largest saving values from the deleted node. This procedure creates a lower bound similar to that of lemma 2.1, but the cost of the constructed 1-tree may of course be larger than the MAST in lemma 2.1 and hence produce a poorer bound for the TSP in question. For examples, see chapter 5. The number of lower bounds, good or bad, constructed in this way will nevertheless be ample. From each matrix C we can construct n saving matrices. From each of these we con construct n different lower bounds by the procedure above.

Moreover, a given lower bound for the TSP in the original cost matrix C automatically creates, by using (2.3), a corresponding upper bound for the TSP in any saving matrix - weighted or not. The sum of the two bounds will be equal to two times the appropriate (weighted) row sum. A new and better lower (larger) bound for the TSP in C will of course create a new and better (smaller) bound in the saving matrix.

2.5.3 Bounds Based on Lagrangean Relaxation

Making a Lagrangean relaxation of a TSP creates a lower bound for any choice of the multipliers. As always when applying Lagrangean relaxation, one has different options for relaxing restrictions. If one chooses to relax both sets of equation in model 1 – that is (1.2) and (1.3) the Lagrangean function that we want to minimise in the asymmetric case can be written as

$$\sum_i \sum_j (c_{ij} - \lambda_i - \mu_j) U_{ij} + \sum_i \lambda_i + \sum_j \mu_j \tag{2.28}$$

where λ_i and μ_j are the multipliers connected with the said equations. Since we want to minimise, and the decision variables are supposed to be binary, we choose a U_{ij} to be equal to one iff the corresponding coefficient in the Lagrangean function is negative. Since we are relaxing equations the multipliers can take on both negative and positive values.

Choosing the multipliers such that $\lambda_i + \mu_j \geq c_{ij}$ for all i, j, the lower bound obtained will be the sum of the multipliers. Such a sum can easily be obtained by solving the following small LP model:

$$\max\left(\sum_i \lambda_i + \sum_j \mu_j\right)$$

$$ST$$

$$\lambda_i + \mu_j \leq c_{ij}$$

$$\lambda_i, \mu_j \geq 0$$

Any optimal solution to the LP model will give the best lower bound possible found by the relaxation above. For further details and elaboration on this, see chapter 4, section 3.

Similarly, one can relax a maximisation problem using any saving matrix as an input matrix, and then getting upper bounds for the complementary problem.

(2.29) $$\sum_i \sum_j \left(s_{ij}^d - \delta_i - \varepsilon_j\right) U_{ij} + \sum_i \delta_i + \sum_j \varepsilon_j$$

where (2.29) is a function we want to maximise. Choosing the multipliers in (2.29) to be $\delta_i = c_{id} - \lambda_u$ and $\varepsilon_j = c_{dj} - \mu_j$ it is easily checked that we obtain the same lower bound for the minimising problem as with (2.28). Hence, the same bounds can be obtained by the first Lagrangean relaxation as with any similar relaxation based on any of the saving matrices.

Each of the saving matrices can give good lower bounds for the TSP problem for instance with the help of lemma 2, the bounds given in subsection 2.5.1, or simple assignments. These bounds will in general be different. A natural question to ask is whether the information about one such bound obtained by a specific saving matrix can be utilised to improve the bounds obtained for an other saving matrix, for example with the help of Lagrangean relaxation? More precisely, let d and l be two nodes and UB_d and UB_l be corresponding upper bounds found for maximising the respective saving matrices, and let Z^* be the optimal solution to the following model:

(2.30) $$\max \sum_i \sum_j s_{ij}^d U_{ij}$$

$$ST$$

(2.31) $$\sum_j U_{ij} = 1 \quad \forall i$$

(2.32) $$\sum_i U_{ij} = 1 \quad \forall j$$

(2.33) $$\sum_i \sum_j s_{ij}^l U_{ij} \leq UB_l$$

If Z^* is strictly less than UB_d and UB_l, we have by (2.19) improved the lower bound for the underlying TSP. The restrictions in the model above consist of a knapsack problem and an assignment problem. One can either try to solve the model as it is – hoping to get better bounds - or one can relax some of the restrictions.

By relaxing (2.33) we get an assignment problem which is easy to solve. However, it turns out that this relaxation does not improve on the already known bounds. This can be seen by the following argument. Let φ_{ass}^{max} and φ_{ass}^{min} be the maximal and minimal assignments respectively in the original cost matrix, and we then suppose that any of the other known bounds are better than these bounds. The Lagrangean function becomes

$$F(\rho) = \sum_i \sum_j \left(s_{ij}^d - \rho s_{ij}^l\right) U_{ij} + \rho UB_k$$

where ρ is a non-negative Lagrangean multiplier. This function can be rewritten as

$$F(\rho) = \sum_i c_{id} \sum_j U_{ij} + \sum_j c_{dj} \sum_i U_{ij} - \rho\left(\sum_i c_{il} \sum_j U_{ij} + \sum_j c_{lj} \sum_i U_{ij}\right) + (\rho - 1)\sum_i \sum_j c_{ij} U_{ij} + \rho UB_l$$

Maximising this function under the assignment restrictions (2.31) and (2.32) can easily be done by inspection when divided into two cases. First, let the multiplier be in the interval [0,1]. Then

$$F^*(\rho) = R_d + K_d - \rho(R_l + K_l) + (\rho - 1)C(\varphi_{ass}^{min}) + \rho UB_l$$

This upper bound corresponds to the lower bound

$$LB_d^*(0 \le \rho \le 1) = \left(LB_l - C(\varphi_{ass}^{min})\right)\rho + C(\varphi_{ass}^{min}) \le LB_L$$

where LB_l corresponds to the lower bound given by (2.19) and the right hand side of (2.32).

In a similar fashion, one can show that if the multiplier is strictly larger than one, it will again be equal or less than LB_l. Hence, in either case no improvement is obtained.

Solving model (2.30) – (2.33) without relaxing any of the restrictions, may however give better, ie lower upper bounds, and as a consequence larger lower bounds for the TSP in question. Suppose that we have found fairly good upper bounds for the TSP measured in the saving matrices based on nodes *d* and *l* by some simple methods, for example one of those described in the previous sub-section. Then taking the upper bound corresponding to node *l* as the right hand side of (2.33), the solution of the model can give better bounds than any of the two known bounds. An example of this will be given in chapter 5.

In principle, we can by using the model (2.30) – (2.33), obtain *nxn* different lower bounds for the TSP in question. Choosing one of the saving matrices in the objective, one can solve the model for *n* different bounds. However, it turns out that most of

these bounds will be equal and that one can only obtain at most n different lower bounds for the original TSP in this way. This can be seen as follows:

Let Z_p^* be the optimal solution to the model when the saving matrix S^p is used in the objective and Z_q^* the optimal solution to the model when the saving matrix S^q is used. In restriction (2.33) we use the same saving matrix in both models with the same upper bound.

Since the models are assignment models with and extra knap-sack restriction, we have by equation (2.8c), that

$$Z_p^* - Z_q^* = 2R_p - 2R_q$$

Each of the optimal solutions are upper bounds for the TSP in their respective saving matrices and as a consequence, give lower bounds in the original cost matrix. These lower bounds are

$$LB_p = 2R_p - Z_p^* \text{ and } LB_q = 2R_q - Z_q^*$$

Hence, the difference between these two lower bounds is

$$LB_p - LB_q = (2R_p - Z_p^*) - (2R_q - Z_q^*) = 0$$

In a similar fashion, better bounds can be found by Lagrangean relaxation, relaxing restriction (2.31) and (2.32), and solving the remaining knapsack problem.

2.6 Savings and Vehicle Routing

If the TSP is extended to m-TSP or VRP one may ask whether equation (2.3) still holds or not. The answer is yes, with the obvious modification that we cannot use a cyclic permutation, but must restrict ourselves to admissible tours, ie we can only use sub-tours containing the depot. Since the depot is supposed to be given á priory, any application of the saving matrices is restricted to the saving matrix given by this depot.

Let $i = 1$ denote the depot and φ' describe the sub-tours. Then $C(\varphi')$ denotes the cost of the sub-tours, and $S(\varphi')$ will be the sum of the corresponding savings. If we have k different sub-tours then $2k$ of the used savings will be zero, since any saving connected to the depot will be zero. The non-zero savings will constitute disjointed chains of nodes and edges instead of one chain containing all the nodes apart from the chosen depot node in the TSP-case. Some of the chains may degenerate to single nodes reflecting the fact that some sub-tours may contain only one customer. Since (2.2) holds, it remains to be shown that $K(\varphi') = 2R_1$. Let i and j be two of the $2k$

nodes connected to the depot node 1. We will then have the following sequence as part of the k sub-tours:

$$....- i - 1 - j -...$$

which gives the following cost elements in K:

$$k_{i1} + k_{1j} = c_{i1} + c_{11} + c_{11} + c_{1j} = c_{i1} + c_{j1}$$

This shows that the sum of the elements used in applying φ' on K does not change the sum. We then obtain equation (2.28), where it is implicitly assumed that the savings in S and the elements of K are based on $i = 1$ as the depot node.

(2.28) $$C(\varphi') + S(\varphi') = K_1 + R_1$$

As in the TSP case, the difference between two solutions of a VRP problem equals the difference between the corresponding savings. (2.28) shows that as in the TSP case, using a saving matrix based on the given depot, is as good as using the original cost matrix.

Equation (2.28) also opens for some of the applications described in the TSP case. This holds for those of the simple heuristics described in sections 1.6 and 2.1 which are adjustable to the VRP case. Note though, that we can no longer use the heuristics in an infinite number of ways since the depot is fixed.

If the cost matrix C is symmetrical, then (2.28) will change to:

(2.29) $$C(\varphi') + S(\varphi') = 2R_1$$

The Fisher - Jaikumar heuristic for VRP

In section 1.5 a short description of VRP can be found. Many of the heuristics for TSP described in chapter 1 can be adapted to VRP with more or less success. Another approach for solving VRP is the specialised heuristic called the generalised assignment heuristic, see Fisher and Jaikumar, 1981. This heuristic basically divides the routing problem into two sub-problems, much in the same way as dispatchers do when they plan routes for the vehicles manually. The first sub-problem consists of assigning a set of customers to each vehicle so that the vehicles' capacities are not violated. These assignments are done in parallel, that is, every vehicle is taken into consideration at the same time. In manual planning, this is done sequentially. The dispatcher fills up one vehicle first, then a second one and so on. The second sub-problem consists of solving a set of TSP, one for each vehicle. This will be reasonably simple since the number of customers assigned to each vehicle in practical applications will not be a very large number, and the TSPs can be solved without side constraints.

A good overall solution depends on how good the assignment part of the heuristic is. In order to obtain good solutions to this part, a selection of so called seed-nodes must be done. Seed-nodes are customer nodes, and one chooses a single seed-node for each vehicle and this vehicle must serve its seed-node. One must take care to select the seed-nodes such that there is only a remote possibility that two seed nodes will be on the same route in an optimal solution. When the seed-nodes have been selected, one calculates the added cost of assigning the other nodes to the same vehicle by formula (2.30). The formula is basically of the same kind as savings, ie: if we serve the seed-node alone with its vehicle, a cost back and forth between the seed-node and the depot is incurred. By assigning another node to the same vehicle an extra cost is incurred, namely:

(2.30) $$c^*_{j,i_v} = c_{j,i_v} + c_{1,j} - c_{1,i_v}$$

where i_v is the selected seed-node for vehicle no. v. The number of vehicles must be decided in advance of the solution of the problem. A generalised assignment problem is now formulated and solved with these new cost elements.

Now, since the saving matrix relative to the given depot can be used as input, one can calculate the loss of saving when serving a node j with a vehicle v attached to the seed node i_v , instead of calculating the increased cost with C as an input matrix. This gives the following formula for the decrease in the savings in the symmetric case. Note that the superscript usually applied to the saving elements is deleted, since only one saving matrix is used.

(2.31) $$s^*_{j,i_v} = 2s_{1,i_v} - (s_{1,i_v} + s_{i_v,j} + s_{j,1}) = -s_{i_v,j} = c^*_{j,i_v} - 2c_{1,j}$$

(2.31) shows that the new cost elements can be found from the old ones, but values can be different and can give quite other solutions compared to the same heuristic applied with C.

2.7 Savings and Special Cases of TSP

In this section some minor extensions to known polynomial classes of TSP will be dealt with. All the extensions are based on the admissible linear transformation that takes place, when a cost matrix is transformed into a saving matrix. This transformation gives a certain degree of freedom, in the sense that entries in a row and the corresponding column in the saving matrix become zero.

2.7.1 Savings and Product Matrices

Let $\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{d}$ be three vectors of dimension n, where the first elements in all vectors are equal to zero.

Define a matrix C, such that

$$c_{ij} = a_{i1} + b_{1j} - d_i d_j$$

It is easily checked that the saving matrix based on node 1 as the depot, will be identical with the product matrix defined by the vector $\boldsymbol{d}$. The maximal cycle in the saving matrix can be found in polynomial time, and hence the minimal cycle in the original cost matrix. The definition of the matrix C reveals that this class of polynomial TSP is a simple linear admissible transformation and the result is a trivial one. However, to identify that a given matrix has this form can be difficult. This difficulty can be circumvented by calculating the saving matrix, and checking whether the saving matrix is a product matrix or not by lemma 1.3.

2.7.2 Savings and the Circulant-TSP

One does not know whether the class of circulant TSP is polynomial or not, but as described in chapter 1, it is easy to find the smallest Hamiltonian path by applying the nearest neighbour procedure to the cost matrix. We will call an *nxn* matrix *almost circulant* iff the entries in the first row and column are constants, and deleting this row and column gives a circulant *(n-1)x(n-1)* matrix. The constants in the first row and column can be different. This rather special class of TSP matrices will be polynomial solvable, since finding the Hamiltonian path can be done in polynomial time and closing the path by including node no. 1 will cost two times the constant, whatever the starting and terminating nodes of the path are.

Now let $\boldsymbol{a}$ and $\boldsymbol{b}$ be two vectors of dimension n *where* the first elements in both vectors are equal, and c_k, $k=2,3,\ldots,n-1$ be $n-2$ non-negative numbers. Define the matrix C to be

$$c_{ij} = \begin{cases} a_{1j} \text{ if } i = 1 \\ b_{i1} \text{ if } j = 1 \\ a_{1j} + b_{i1} - c_k \ \forall i, j \geq 2 \text{ and } (j-i) = k(\text{mod}(n-1)) \end{cases}$$

It is easily shown that the saving matrix based on node 1 as a depot, will become *almost circulant* with zeros in the first row and column. The entries on the main diagonal do not concern us. We then find the maximal Hamiltonian path in the saving matrix, and this can of course then be done by applying the farthest neighbour procedure to the circulant part of the matrix. The corresponding Hamiltonian cycle will then be the maximal in the saving matrix, and the minimal one in the original cost matrix.

2.7.3 Savings and the Small TSP

With a *large* matrix C with distinct values, we will understand a matrix where the entries are governed by two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ with distinct elements, such that

$c_{ij} = \max\{a_i, b_j\}$. As in subsection 1.7.2 we can assume, without loss of generality and for simplicity of notation that $a_1 \geq a_2 \geq ..... \geq a_n$. Let d_i be the *ith* largest of the *2n* distinct elements of the two vectors. Further, let $D = \{d_1, d_2, ...d_{n-1}\}$ and $d = \sum_{i=1}^{n-1} d_i$.

Theorem 2.4

Let S be a large matrix. The length of the largest Hamiltonian path in S is d iff one of the following four conditions holds:

S1) *For the same node i, both* a_i and b_i *are in D*

S2) $D = \{a_1, a_2, ..., a_{n-1}\}$

S3) $D = \{b_1, b_2, ..., b_{n-1}\}$

S4) $D = \{a_1, a_2, ...a_k, b_{k+1}, b_{k+2}, ...b_{n-1}\}$ for some k, $1 \leq k \leq n-2$

Proof:
The proof follows very closely that of Gabovich, which can be found in Lawler et alt., 1985.

Suppose that S2 holds. Then the entries in row *i* in the matrix *C* will be a_i, since the elements of the ***a*** vector will dominate those of the ***b*** vector. Hence, the matrix *C* is a constant TSP matrix. By corollary 1.1 the largest Hamiltonian path in such a matrix is $\sum_i a_i - \min_i\{a_i\} = d$. If S3 holds, a similar result is obtained. If S4 holds, the arcs in *D* can be organised into a path with cost *d*. Since this value is the largest possible for combining *n-1* arcs in any way, this will be a maximal path.

Suppose that S1 holds. Define the following subsets of the index set *i:*

$$D_0 = \{i | a_i \notin D \wedge b_i \notin D\}$$
$$D_a = \{i | a_i \in D \wedge b_i \notin D\}$$
$$D_b = \{i | a_i \notin D \wedge b_i \in D\}$$
$$D_2 = \{i | a_i \in D \wedge b_i \in D\}$$

By assumption, $D_2 \neq \varnothing$. Let $E = \{e_1, e_2, ..., e_n, e_{n+1}, ..., e_{2n}\}$ be the elements of the two vectors ordered in descending order. If $i \in D_2$, then a_i and b_i will be among the *n* first elements in *E*. Then there must exist a *j* such that a_j and b_j belong to the *n* last elements in *E*, that is $j \in D_0$ and vice versa. In other words, the cardinalities of D_0 and D_2 must be the same. We then make a Hamiltonian cycle by starting with any node in D_2, then go to any node in D_a, and so visiting every node in D_a in any order, Continue to any node in D_0, then to any node in D_b and visit every node in D_b in any order. Then return to a node in D_2 (if any left) and then go back and forth between the

remaining nodes in D_0 and D_2 until every one is visited and then return to the starting node. The cost of this cycle will be $d' = \sum_{i=1}^{n} d_i$. Since all the entries in the cost matrix are distinct each arc of a cycle must have a different value, and the values in D are the largest one, and the cost of any cycle must be at least d'. Hence the constructed cycle above is an optimal one. From this cycle we delete the smallest one and obtain a Hamiltonian path with cost d.

Now, suppose that neither S1, S2, S3 nor S4 hold. Then D will contain at least one b followed by some a. A path with cost d must have arcs corresponding precisely to the costs in D, therefore in such a path some arcs have a cost corresponding to some a values, and some that correspond to b values. Hence somewhere along the path. a b_s must be followed by an a_t, but that is not possible unless $s = t$, which is impossible by a.

QED

Note that a matrix made up from the sum of a constant TSP matrix and any polynomial solvable matrix, will again be polynomial solvable, even if the sum does not have the necessary properties. However, in order to realise that a matrix consists of such a sum cannot always be done by inspection alone and some kind of transformation is necessary. One such transformation can be by taking one of the saving matrices. Then deleting the row and column corresponding to the chosen depot, one may check whether the remaining matrix is large or not. If it is large, one finds the maximal Hamiltonian path by the previous theorem. Adding the two remaining arcs in order to obtain a cycle will give no extra costs, and this cycle will be the shortest Hamiltonian cycle in the original cost matrix. We then have the following corollary:

Corollary 2.4

Let A be any matrix such that for some node d, the corresponding saving matrix is a large matrix with row and column d deleted, can be solved in polynomial time.

2.7.4 Savings and Pyramidal Tours

To find an optimal cycle among the pyramidal cycles one can follow a dynamic programming scheme shown in Lawler et alt., 1985. To ensure that an optimal tour is pyramidal, certain algebraic structures are imposed on the cost matrix. Such structures seem in most cases to involve at least four cost elements, and sometimes as many as twenty as is the case for asymmetric Demidenko matrices.

This scheme can easily be adapted to find a maximal pyramidal Hamiltonian *path* starting in node 1 and going through ascending indices up to node n and then through descending indices ending in some node, simply by extending the matrix by an artificial node, and finding the shortest pyramidal cycle by taking the original matrix and multiply each entry by minus one.

Monge matrices

Let C be a Monge matrix as defined in 1.7.7. The algebraic structure in C is preserved if the matrix is transformed into a saving matrix. This can be seen from the following simple calculations:

$$s_{ij}^d + s_{rs}^d = c_{id} + c_{dj} - c_{ij} + c_{rd} + c_{ds} - c_{rs} \geq c_{id} + c_{dj} + c_{rd} + c_{ds} - (c_{is} + c_{rj}) = s_{is}^d + s_{rj}^d$$

where inequality is obtained since C is Monge. Hence, in this respect nothing is achieved by transforming the original matrix to a saving matrix.

On the other hand, we can start with a matrix which is *not* Monge, but which will become Monge as soon as the first row and column is deleted. Hence the corresponding saving matrix based on node 1 as the depot node will be Monge when the first row and column in the saving matrix is deleted. This row and column contain only zeros. Hence, by finding the longest pyramidal Hamiltonian *path* in the saving matrix will be equivalent to finding the TSP in the original cost matrix. By the introductory remark in this sub-section this can be done in polynomial time.

In the same simple way the structures of Van der Veen matrices, asymmetric and symmetric Demidenko matrices, Supnic matrices, Kalmanson matrices, and generalised distribution matrices, are preserved when the original cost matrices are transformed to any saving matrix. The classes can then be extended by adding an arbitrary row and column as described for the Monge matrices.

2.7.5 A new Class of Polynomial Solvable TSP

In some of the different classes of TSP treated so far, the underlying matrix can be described with the help of two vectors. This is the case with the constant-TSP, the product matrices and the Brownian matrices. The new class is partly described in a similar way. Let $\boldsymbol{a}$ and $\boldsymbol{b}$ be two vectors both with dimension n and with zero as their first element. The first row in the matrix consists of the elements in the vector $\boldsymbol{a}$ and the first column of the vector $\boldsymbol{b}$. Each remaining entry below the main diagonal is constructed in the constant-TSP way from the two vectors. The entries on the main diagonal are zero. The remaining entries above the main diagonal can be any numbers. More formally, let the matrix be denoted C. Then the entries are defined as

$$c_{ij} = \begin{cases} a_j \ \forall j, i = 1 \\ b_i \ \forall i, j = 1 \\ 0 \quad if \ i = j \\ b_i + a_j \ \ if \ i > j; i, j \neq 1 \\ any \ number \ if \ i < j; i, j \neq 1 \end{cases}$$

In order to prove that this class of matrices is polynomial solvable, one calculate the saving matrix for the matrix based on node 1 as the depot. As we have noted before the first row, the first column, and the main diagonal in the saving matrix will then consist of zeros. In general the saving values will be

$$s_{ij} = c_{i1} + c_{1j} - c_{ij}$$

For the matrix in question, we get by definition that

$$s_{ij} = b_i + a_j - c_{ij}$$

Now, let $i > j$, and $j > 1$, that is, for all the entries below the diagonal. Then the corresponding saving values become:

$$s_{ij} = b_i + a_j - (b_i + a_j) = 0$$

The saving values above the main diagonal, apart from the first row can be different from zero and do not concern us. Hence, the saving matrix is an upper triangular matrix and we can find the maximal TSP in this matrix. This cycle will then correspond to minimal cycle in the original cost matrix.

Note, the class of matrices defined above, constitutes a vector space.

2.8 Savings and Edges

Finding optimal solutions to TSP can be difficult when the number of nodes becomes large, whatever exact methods one uses. Having á priori knowledge about edges or arcs that will be in an optimal solution, or similar knowledge about edges or arcs not in any optimal solution, will in most cases make the burden of computation less. In the next two sub-sections these aspect are dealt with.

2.8.1 Edges that are in an Optimal Solution

By *an optimal edge (arc)* we understand an edge (arc) *(i,j)* that is a part of an optimal cycle. Warren, 1992, offers two theorems on how to find such edges or arcs. The first theorem deals with the general case and is for convenience repeated below.

Theorem 2.5

Let σ be a cyclic permutation on the n nodes such that $\sigma(i) \neq j$. If either
$c_{fj} + c_{ei} + c_{i,\sigma(i)} \geq c_{e,\sigma(i)} + c_{fi} + c_{ij}$ *or* $c_{i,\sigma(i)} + c_{f,j} + c_{j,\sigma(j)} \geq c_{f,\sigma(j)} + c_{j,\sigma(i)} + c_{ij}$
where $e = \sigma^{-1}(i)$ *and* $f = \sigma^{-1}(j)$
then (i,j) is an optimal arc.

The proof is based on a simple 3-exchange procedure, which preserves the directions of the arcs not involved in the exchange procedure. The second theorem deals with the symmetric case.

Theorem 2.6

Let σ be a cyclic permutation on the n nodes such that $\sigma(i) \neq j$ and $\sigma(j) \neq i$. If either $c_{jj} + c_{i,\sigma(i)} \geq c_{ji} + c_{\sigma(i),\sigma(j)}$ or $c_{ei} + c_{jj} \geq c_{ij} + c_{ef}$ where $e = \sigma^{-1}(i)$ and $f = \sigma^{-1}(j)$ then (i,j) is an optimal arc.

In this case the proof is based on a simple 2-change procedure.

Note that optimal arcs or edges can be found in time $O(n^4)$. Note also, that to find more than one optimal edge or arc at a time does not help very much. If one does know two such arcs one cannot utilise this fact unless one in addition knows that both arcs are on the same cycle. However, knowing that a certain edge or arc - say *(i,j)* - is optimal, we can reduce the dimension of the TSP by one by deleting row *i* and column *j*, setting the cost of *(j,i)* to infinity, rearranging the numbering and solve a TSP of dimension *(n-1)* to optimality. This optimal solution can then be lifted back to give the optimal solution in the original matrix.

If one has found an optimal edge in a symmetric instance of TSP, one can use this information to find new lower bounds for the problem in the same way as was done in (2.27) The difference is that we now can delete two nodes from the graph, namely the two nodes given by the optimal edge, say *(i,j)* and all edges attached to these two nodes Then, first, making MIST on the remaining nodes $N - \{i, j\}$, we then add the edge again and the smallest remaining edge attached to *i* and the smallest remaining edge attached to *j*. The resulting 1-tree will be a lower bound for TSP.

One question remains. Will the transformation to savings give different optimal arcs when applied to the two theorems above? The answer to this question is negative. It is easily shown that by replacing the cost elements in the premises of the theorems by corresponding saving elements for some choice of the depot *d*, and changing the direction of the inequalities, will give exactly the same result.

2.8.2 Edges or Arcs that are not in any Optimal Solution

An arc is called *non-optimal* for a TSP if no optimal solution contains it. A pair of arcs is called a *non-optimal pair* for a TSP if no optimal solution contains the pair. A triple of arcs is called a *non-optimal triple* if no optimal solution contains the triple. We will start this sub-section considering non-optimal pairs, and to the best of my knowledge this concept has not been treated before, or non-optimal triples either.

Theorem 2.7

Let C be a symmetric matrix and (i,j) and (s,t) two edges, all nodes different. If $c_{ij} + c_{st} > c_{is} + c_{jt}$, then (i,j) and (s,t) is a non-optimal pair.

Proof: The proof is a simple 2-change procedure. Let σ be a cycle containing the two edges. By definition they are not neighbours in the cycle. Delete the two edges and

replace them by *(i,s)* and *(j,t)*. The new cycle has a cost that is strictly less than the cost of σ. Clearly, the pair must be non-optimal.

QED

Note, that given a non-optimal pair, one of the edges can perfectly well be a part of an optimal solution, but not both. Hence, we can add to any model for TSP for any set of non-optimal pairs, constraints of the type:

$$U_{ij} + U_{st} \leq 1$$

Theorem 2.8

Let C be a matrix.

If (i,j), (j,k), and (s,t) are three arcs, all nodes different, and $c_{ij} + c_{jk} + c_{st} > c_{ik} + c_{si} + c_{it}$ *then the three arcs comprise a non-optimal triple.*

If (i,j), (k,l) and (p,q) are three arcs, all nodes different, and $c_{ij} + c_{kl} + c_{pq} > c_{il} + c_{kq} + c_{pj}$ *then the three arcs comprise a non-optimal triple.*

Proof: The proof is a simple 3-change procedure that preserves the direction of the arcs not involved in the exchange procedure. Let σ be a cycle containing the three arcs. In the first case two of the arcs are neighbours and the third *(s,t)* is not adjacent to any of the other two. Delete the three arcs and replace them by *(i,k), (s,i,) and (i,t).* The new cycle has a cost that is strictly less than the cost of σ. Clearly, the triple must be non-optimal. In the second case none of the three arcs are adjacent. Delete the three arcs and replace them by *(i,l), (k,q)* and *(p,j)* respectively. The new cycle has a cost that is strictly less than the original one. The result follows.

QED

The proofs of the two last theorems follows very closely the proofs given by Warren for theorem 2.x and 2.y, only the objective is the opposite, finding non-optimal arcs rather than optimal ones.

Having identified a set of non-optimal triples, one can in a model for solving TSP add restrictions of the following type for every triple:

$$U_{ij} + U_{jk} + U_{st} \leq 2$$

A natural question to ask is whether the two theorems above can give more information if we use saving matrices instead. However, it turns out that this is not the case. The conditions of the theorems are full-filled in the original matrix if and only if the same is true for the saving matrices.

The two previous theorems do not provide single non-optimal edges or arcs. Such single arcs can be found by a technique provided by Jonker and Volgenant, 1982. They offer two theorems. Only one of them will be treated here. However, some notation is necessary.

Let F denote the set of non-optimal arcs identified at a certain point in the solution process. F may be empty or non-empty. Further, let

$$\Delta_{ij}^{k}(C) = c_{ij} - (c_{ik} + c_{kj}) \tag{2.32}$$

for nodes k different from i,j. Note that if $\Delta_{ij}^{k}(C) \geq 0$ for all choices $i,j,$ *and* k the triangle inequality holds for the cost matrix C. Let

$$\mu_{ij}^{k}(C) = \max_{p,q}\left\{\Delta_{ij}^{k}(C) \middle| p \neq i; q \neq j; (p,q) \neq (j,i); (p,k),(k,q) \notin F\right\} \tag{2.33}$$

Hence, in order to determine $\mu_{ij}^{k}(C)$, only arcs (p,q) that can be contained in a TSP solution together with (i,j) are considered. If $\mu_{ij}^{k}(C)$ does not exist, then the arc (i,j) is non-optimal. What will happen if $\mu_{ij}^{k}(C)$ does exist? Jonker and Volgenant offer the following theorem:

Theorem 2.9

If there exists a node k, $k \neq i, j$ *such that* $\Delta_{ij}^{k}(C) > \mu_{ij}^{k}(C)$, *then the arc (i,j) is non-optimal.*

The proof is based on a simple 3-change procedure that preserves the direction of the arcs not involved in the exchange procedure and the fact that any TSP solution that is not 3-optimal is not optimal.

To identify whether or not a given node k is non-optimal and checking the premises of the theorem takes $O(n^2)$ time. A different order for the values of k may influence which and how many non-optimal arcs are identified. Hence, there may be some virtue one in applying or more of the saving matrices. Transforming to a saving matrix we get:

$$\Delta_{ij}^{k}(S^d) = (c_{kd} + c_{dk}) - \Delta_{ij}^{k}(C) \tag{2.34}$$

Hence, one criterion can be found from the other, but the effect of applying a saving matrix can be different. For example, even if the original cost matrix obeys the triangle inequality, the corresponding saving matrices will not. So, redefining (2.xx) to

$$\mu_{ij}^{k}(C) = \min_{p,q}\left\{\Delta_{ij}^{k}(C) \middle| p \neq i; q \neq j; (p,q) \neq (j,i); (p,k),(k,q) \notin F\right\} \tag{2.35}$$

From theorem 2.9 we get the following corollary:

Corollary 2.5

Let S^d be any saving matrix of the original cost matrix. If there exists a node k, $k \neq i, j$ such that $\Delta_{ij}^k(S^d) < \mu_{ij}^k(S^d)$, then the arc (i,j) is non-optimal.

2.9 TSPs with Identical Costs for every Cycle.

If one has a constant-TSP instance, the cost of every Hamiltonian cycle is the same. Given two different instances of a constant-TSP, the cost of a Hamiltonian cycle in one single instance can be the same as that of the other, even if each entry in the cost matrices is different. This will happen if the sums of the elements of the two pairs of vectors describing the cost matrices are equal. The same Hamiltonian cycle calculated in two different equally sized cost matrices will of course, in general, have different costs. An interesting question to ask then, is what kind of relations must there be between two different matrices *A* and *B* if the cost of every Hamiltonian cycle shall be the same in the two matrices? Ie, what kind of properties must the two matrices have, if

$$\forall \varphi, A(\varphi) = B(\varphi) \tag{2.36}$$

Now let $S^d(A)$ denote the saving matrix based on the matrix *A*, and node *d* as the depot, then $S^d(A)(\varphi)$ denotes the cost of the Hamiltonian cycle φ calculated in this matrix, and finally $s_{ij}^d(A)$ the entries in $S^d(A)$.

The following lemma gives a characterisation of matrices obeying (2.36)

Lemma 2.2

Let A and B be two nxn matrices with zeros on the main diagonal and k a constant. Then it follows that:

$$\forall \varphi, A(\varphi) = B(\varphi) \text{ iff } \exists d, \forall i, j \neq d; s_{ij}^d(A) - s_{ij}^d(B) = k \text{ and}$$
$$\left(R_d(A) + K_d(A)\right) - \left(R_d(B) + K_d(B)\right) = k(n-2)$$

Proof:

Suppose that $\forall \varphi, A(\varphi) = B(\varphi)$. Then by (2.3) it follows that $\forall \varphi, S^d(A)(\varphi) - S^d(B)(\varphi) = \left(R_d(A) + K_d(A)\right) - \left(R_d(B) + K_d(B)\right)$ for any choice of the depot node *d*. For each cycle φ the left-hand side of these equations consists of $2n$ parts and the right-hand side is the same for every φ, hence a constant. If i or j are equal to the chosen depot node *d*, $s_{ij}^d(A) = s_{ij}^d(B) = 0$. Hence, for the rest of the left-hand side it follows that

(2.37) $$\forall \varphi, \sum_{\substack{i=1 \\ i,\varphi(i)\neq d}}^{n} \left(s^d_{i,\varphi(i)}(A) - s^d_{i,\varphi(i)}(B)\right) = \left(R_d(A) + K_d(A)\right) - \left(R_d(B) + K_d(B)\right)$$

constitutes a set of linear equations with *(n-1)!* equations and *2(n-1)(n-2)* variables. The left-hand sides consist of *(n-2)* differences between two variables and all the right-hand sides are equal. Now, setting

(2.38) $$s^d_{i,\varphi(i)}(A) - s^d_{i,\varphi(i)}(B) = x_{i,\varphi(i)} + \frac{\left(R_d(A) + K_d(A)\right) - \left(R_d(B) + K_d(B)\right)}{n-2}$$

into (2.37) gives

(2.39) $$\forall \varphi, \sum_{\substack{i=1 \\ i,\varphi(i)\neq d}}^{n} x_{i,\varphi(i)} = 0$$

constitutes a homogenous system of linear equations. The trivial solution is always a solution to such a system, and so by (2.38), it follows that

(2.40) $$s^d_{i,\varphi(i)}(A) - s^d_{i,\varphi(i)}(B) = \frac{\left(R_d(A) + K_d(A)\right) - \left(R_d(B) + K_d(B)\right)}{n-2}$$

By setting $k = \frac{\left(R_d(A) + K_d(A)\right) - \left(R_d(B) + K_d(B)\right)}{n-2}$ we have the first part of the right-hand side of the equivalence in the lemma. The second part follows immediately.

Now suppose that

$$\exists d, \forall i, j \neq d; s^d_{ij}(A) - s^d_{ij}(B) = k \text{ and } \left(R_d(A) + K_d(A)\right) - \left(R_d(B) + K_d(B)\right) = k(n-2).$$

We then have that

$$A(\varphi) = R_d(A) + K_d(A) - S^d(A)(\varphi) = R_d(B) + K_d(B) + k(n-2) - \sum_{i=1}^{n} s^d_{i,\varphi(i)}(A) =$$

$$R_d(B) + K_d(B) + k(n-2) - \sum_{i=1}^{n} \left(s^d_{i,\varphi(i)}(B) + k\right) =$$

$$R_d(B) + K_d(B) + k(n-2) - \sum_{i=1}^{n} \left(s^d_{i,\varphi(i)}(B)\right) - k(n-2) = R_d(B) + K_d(B) - S^d(B)(\varphi) = B(\varphi)$$

QED

Lemma 2.2 shows that in a complete graph we have infinitely many matrices which give the same costs when calculating the costs of the Hamiltonian cycles. Now, let for any matrix let A, $I_A = \{B | \forall \varphi; A(\varphi) = B(\varphi)\}$. Note that if A is in a polynomial class then each $B \in I_A$ can be solved polynomial as well, that is, we use the polynomial instance as input to our TSP. Below are some questions asked, but not answered.

Question 1

Given an arbitrary matrix, will this matrix belong to a set I_A such that A is polynomial?

The answer to this question is probably "no". If it turns out to be "yes", then every matrix can be reduced to a polynomial case, and hence every matrix can in principle be solved polynomial, which seems to be a bit far-fetched and leads to the conclusion that $P = NP$. On the other hand, if the answer turns out be "yes", finding the polynomial class or identifying such a class, may be as difficult as solving the original TSP.

Question 2

If A is in a polynomial class, will every $B \in I_A$ be in the same class?

If this is the case, the construction in lemma 2.2 does not lead to any new information.

Question 3

Given the I_A, will some $B \in I_A$ be easier to solve than others for example in terms of better lower bounds, easier or fewer facets? If "yes", how is such easy matrices identified?

Question 4

How do heuristics perform on the different members of I_A?

Since the different saving matrices for all the matrices in I_A are identical apart from a constant, most heuristics will probably find the same cycles independently, regardless of which saving matrix we use as an input to the heuristic. On the other hand, the original cost matrices can be quite different, and yield different cycles when exposed to a given heuristic.

Chapter 3

In the previous chapter the cost matrix of a TSP instance was divided into variants of saving matrices and a constant-TSP matrix. These sorts of transformations are all linear admissible transformations. In this chapter we will look at other ways to decompose the cost matrix for a TSP instance. These decompositions will not be linear admissible transformations.

In his article from 1990 Burkhard briefly discusses what he calls the *approximation problem*:

"Let a cost matrix A be given. We want to approximate A by a cost matrix A^* such that

- The TSP with cost matrix A^* is polynomial solvable, and
- The optimal solution with respect to A^* is not "too far" away from the optimal solution with respect to the cost matrix A.

A lot of questions arise in connection with this (latter) problem: Which special case should be chosen? How can we measure the "proximity"? Are there (deterministic or probabilistic) performance bounds for the approximation? Can classes of problems be characterized which are well-suited for an approximation? And last, but not least, computational studies are needed for such an approximation."

Further, he remarks that "... special classes and heuristics have only been treated separately up to now and no study combining these two approaches is available. But it seems a promising task to base heuristics on special cases."

Since this article was published, many new classes of special cases for TSP have been found in addition to those treated in Lawler et alt, 1985. The latter ones certainly inspired many of the articles referred to in section 1.7, but to the best of my knowledge Burkardts suggestion that a general TSP instance could be approximated by some well-solved case of TSP seems not to have been followed up in any degree. Burkhard's suggestion is briefly mentioned by Warren, 1994, but he does not follow up this line of research. He concentrates on giving a brief, but very good overview of special classes. The only case known to me using an approximation approach and preceding that of Burkard, is Burkov and Rubinstein, 1983. They establish sufficient conditions for the existence of a Hamiltonian cycle, find a solvable TSP class connected with such graphs and use this solvable class to approximate general TSPs.

In this chapter, an approach other than that used by Burkov and Rubinshtein will be followed, may be more in the line with what Burkard probably had in mind. Different

decomposition schemes will be suggested and used for approximations for TSP, both STSP and ATSP.

We will consider mainly how the so-called spectral theorem in linear algebra can be used to decompose a square matrix into different sets of different and simpler matrices, ie simpler to solve.

The literature on TSP is vast. The core of the problem is of course the combinatorial nature of the problem. However, as we have seen, the structure of the cost matrix can play an essential part when one tries to solve an instance of TSP. The fact that the cost matrix of an instance of TSP, is a quadratic, usually non-negative matrix, makes it a bit surprising that standard linear algebra does not seem to play any important part in the literature about TSP, or other combinatorial problems related to TSP. There exists of course some important exceptions as we have seen in chapter 1, notably the results on constant-TSP by Berenguer, Lenstra and Rinooy Kan, which explicitly use techniques and concepts from linear algebra. Another important application is a result for quadratic assignment problems, where good upper and lower bounds for QAP are established with the help of eigenvalues related to appropriate cost matrices for the problem, see Finke et alt., 1987, Hadley et alt., 1989 and Hadley et alt., 1992.

3.1 Decomposition of Symmetric TSP and the Spectral Theorem

In this section we will establish a decomposition of symmetric instances of TSP. The decomposition will consist of a sum of polynomial solvable symmetric matrices, using concepts and techniques from linear algebra. The concepts and techniques used are standard, with one exception. These can be found in almost any introductory textbook on linear algebra, eg Anton and Rorres, 1987. For convenience, the main results are stated as four theorems below.

Theorem 3.1

If A is a quadratic matrix with real entries then the following are equivalent

a) λ *is an eigenvalue of A*
b) *There is a non-zero real vector* $\mathbf{x}$ *in* R^n such that $Ax = \lambda x$
c) *The system of equations* $(\lambda I - A)\mathbf{x} = \mathbf{0}$ *has non-trivial solutions*
d) λ *is a real solution to the characteristic equation* $\det(\lambda I - A) = 0$

where I is the identity matrix.

Theorem 3.2

a) *The characteristic equation of a symmetric matrix with real entries has only real roots.*
b) *If an eigenvalue* λ *of a symmetric matrix is repeated k times as a root of the characteristic equation, then corresponding eigenspace is k-dimensional.*

Definition 3.1

A square matrix A is called orthogonally diagonalizable if there is an orthogonal matrix P such that $P^{-1}AP$ is diagonal.

Theorem 3.3

If A is a nxn square matrix, the following are equivalent:

a) A is orthogonally diagonalizable
b) A has an orthonormal set of n eigenvectors
c) A is symmetric

Since the cost matrix of TSP is supposed to be symmetrical and all its elements are real numbers, then all the eigenvalues of the matrix will be real, and the corresponding eigenvectors will be real as well. Let λ_k, $k = 1,2,....,n$ denote these eigenvalues. Further, we denote the eigenvectors corresponding to the eigenvalues by:

$$\vec{V}_k = [v_{k1}, v_{k2}, ..., v_{kn}] \text{ for } k=1,2,...,n.$$

The set of eigenvectors constitutes an orthonormal basis for the *vector space* R^n. Let D be the matrix defined by $D = [d_{ij}]$ where $d_{ij} = 0$ when $i \neq j$ and $d_{kk} = \lambda_k$. Further, let

$$P = \left[\vec{V}_1^T \middle| \vec{V}_2^T \middle| ..... \middle| \vec{V}_n^T \right]$$

where A^T denotes the transpose of any matrix. Since the eigenvectors are orthonormal the matrix P will be orthonormal as well, and hence the inverse matrix $P^{-1} = P^T$.

A standard result in linear algebra is then the so-called ***spectral theorem***

Theorem 3.4

For any symmetric real matrix C, there exists real eigenvalues and eigenvectors such that

(3.1) $$C = PDP^T$$

From (3.1) we get (3.2)

(3.2) $$c_{ij} = \sum_{k=1}^{n} \lambda_k v_{ki} v_{kj} \quad \forall i, j$$

Lemma 3.1

Let φ be any cyclic permutation in the cost matrix C. Then the cost of the Hamiltonian cycle will be:

(3.3) $$C(\varphi) = \sum_{k=1}^{n} \lambda_i < \vec{V}_k, \vec{W}_k >$$

where $\vec{W}_k = \left[v_{\varphi(i)}\right]$ and $< \vec{V}_k \vec{W}_k >$ is the standard scalar product of the two vectors and the set of vectors $\vec{V}_k$ forms an orthonormal basis for the vector space R^n.

Proof:

Let $C(\varphi) = \sum_{i=1}^{n} c_{i,\varphi(i)}$. Then substituting (3.2) into this expression gives

$$C(\varphi) = \sum_{i=1}^{n}\sum_{k=1}^{n} \lambda_k v_{ki} v_{k,\varphi(i)} = \sum_{k=1}^{n} \lambda_k \sum_{i=1}^{n} v_{ki} v_{k,\varphi(i)}$$

Define $\vec{W}_k$ as in the lemma and (3.3) follows. The elements in each of the vectors $\vec{W}_k$ are created by the same cyclic permutation of the elements of the vectors $\vec{V}_k$. Since φ is a cyclic permutation and the set of vectors $\vec{V}_k$ constitutes an orthonormal basis, it follows that:

$$< \vec{W}_k, \vec{W}_k >= \sum_{i=1}^{n} v_{k_{\varphi(i)}}^2 = \sum_{i=1}^{n} v_{k_i}^2 = 1$$

and for different indices *k* and *l*, one gets

$$< \vec{W}_k, \vec{W}_l >= \sum_{i=1}^{n} v_{k,\varphi(i)} v_{l,\varphi(i)} = \sum_{i=1}^{n} v_{k,i} v_{l,i} = \ < \vec{V}_k, \vec{V}_l > \ = 0$$

QED

Note that the scalar products in (3.3) are always in the interval [-1,1]

In some cases the set of eigenvalues can contain identical eigenvalues. This will happen if the characteristic equation, that is the determinant $|C - \lambda I| = 0$, has multiple solutions. If this multiplicity is *s* for some eigenvalue λ_k, then one can find an orthonormal basis for the eigenspace corresponding to λ_k. This eigenspace will have dimension *s*. Together with the other distinct eigenvalues and their corresponding eigenvectors, we can repeat the arguments above. Further, the values of the eigenvalues can take any values, and the values of the elements of the eigenvectors can take on any value in the interval [-1, 1]

Now, define *n* different symmetric product matrices by

(3.4) $$T_k = \left[v_{ki} v_{kj}\right]_{nxn} \quad \forall k$$

The following corollary arises directly from (3.2)

Corollary 3.1

Let C be any symmetric and quadratic matrix. C can then be decomposed into a weighted linear sum of n different symmetric product matrices

$$C = \sum_{k=1}^{n} \lambda_k T_k$$

where λ_k is the k-th eigenvalue of C.

Hence, any symmetric instance of TSP can be transformed to a sum of *n* polynomial solvable cases of TSP. This of course does not mean that the original TSP is polynomial, since the optimal solutions for two different components do not need to be identical. This observation leads to the next section.

3.2 A New Class of Polynomial Solvable STSP

In order to describe this new class, it is convenient to introduce the following concept.

Definition 3.2

A set of m linear independent vectors $\vec{A}_k = [a_{ki}], i = 1,2,\ldots..,n$, k=1,2,...,m is called homogenous ordered iff $i > l$ then $\forall k, a_{ki} \geq a_{kl}$. The set of corresponding symmetric product matrices is called a homogenous ordered set of matrices.

Let $\vec{A}_k = [a_{ki}], i = 1,2,\ldots..,n$ be *m* homogenous ordered vectors. Then the symmetric matrix

$$C = \sum_{k=1}^{m} \left[a_{ki} a_{kj}\right]$$

is polynomial solvable since all the symmetric product matrices obtain their optimal solution with the same cycle.

The choice of vectors can be restricted to a linear independent set of vectors, because if $\vec{B} = \alpha \vec{A} = \alpha[a_i]$, for some scalar α the corresponding matrix *B*, will be $B = \alpha^2 \left[a_i a_j\right]$ and hence the sum of *A* and *B* can be replaced by the symmetric product matrix $(1+\alpha^2)\left[a_i a_j\right]$

Finding polynomial algorithms for special cases of TSP can be difficult. Once identified, however constructing an instance of such a special case is usually easy, as it is in this case. On the other hand, identifying whether an arbitrarily given square matrix corresponds to a special case or not can in itself be difficult and some fairly simple criteria are needed. Such criteria are known for some special cases of TSP as for example for symmetric product matrices. One such criterion is given in sub-section 1.7.5. However, two or more linear independent vectors with the property above, will not necessarily create a new symmetric *product* matrix. Another way of stating this is to say that the set of such matrices does not constitute a vector space.

In order to identify whether a given symmetric matrix is of the said kind, we need another criterion. However, it seems not to be a straightforward way to identify such criterion.

An approximation heuristic for positive symmetric matrices:

At the beginning of this chapter Burkhardt was cited, indicating the possibilities for combining heuristics with classes of polynomial solvable TSP instances, to approximate a given TSP. Here we will use this idea and approximate a symmetric, positive TSP, with symmetric product matrices that are homogeneously ordered.

Step 1:

Solve the following LP-model

$$\max \sum_{i=1}^{n} x_{1_i}$$

$$ST$$

$$x_{1_i} + x_{1_j} \leq \ln c_{ij} \qquad \forall i, j; i \neq j$$

All variables positive

Let $\overrightarrow{a_1}$ be the vector where $a_{1_i} = e^{x_{1_i}}$ for all positive variables and zero otherwise. Define the matrix A_1 as the corresponding symmetric product matrix. Let $C_2 = C - A_1$. This matrix is non-negative. Go to step *k*.

Step k:

Solve the following LP-model

$$\max \sum_{i=1}^{n} x_{k_i}$$

$$st$$

$$x_{k_i} + x_{k_j} \leq \ln c_{k_{ij}} \qquad \forall i, j; i \neq j \text{ and } c_{k_{ij}} > 0$$

All variables positive

The ordering of x_{k_i} shall be the same as for x_{1_i}

Let $\overrightarrow{a_k}$ be the vector where $a_{k_i} = e^{x_{k_i}}$ for all positive variables and zero otherwise. Define the matrix A_k as the corresponding symmetric product matrix. Let $C_{k+1} = C_k - A_k$. This matrix is non-negative. If there still exists positive entries in C_{k+1} go to step *k+1*. If not, stop.

The heuristic will create a sequence of non-negative symmetric product matrices with the same optimal cycle. The sum of the matrices will be closer and closer to the original cost matrix and hence create better and better lower bounds. The optimal cycle for the product matrices will of course give an upper bound when applied on *C*.

3.3 Non-negative Matrices and the Spectral Theorem

In the first section symmetrical matrices and the spectral theorem were treated. At least, for most applications of TSP the entries in the cost matrix will be non-negative numbers, or can be made non-negative by adding some constant to all the entries without changing the optimal solution. Introducing the non-negativity into the spectral theorem opens for some further possibilities.

In order to obtain the main result in this section we need some more definitions.

Definition 3.3

A quadratic non-negative matrix A is primitive iff there exists a positive integer k such that $A^k > 0$.

Definition 3.4

A matrix P is called a permutation matrix of the identity matrix I, iff P can be obtained from I by interchange rows and or columns in I.

Definition 3.5

A quadratic matrix A is said to be reducible iff there exists a permutation matrix P such that

$$P^T AP = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix}$$

where A_{11} and A_{22} *are quadratic matrices of smaller sizes than A.*

Definition 3.6

By the spectral radius of A, we will understand

$$\rho(A) = \max_i \{|\lambda_i|\}$$

where λ_i *denotes the eigenvalues of A.*

Lemma 3.2

Let C be the cost matrix for any instance of TSP, such that $c_{ii} = 0$ and $c_{ij} > 0, i \neq j$. *Then the matrix C is irreducible.*

Proof:

By ordinary matrix multiplication one gets that $C^2 > 0$, that is, C is primitive. Any positive matrix is per definition irreducible. Hence, C^k is irreducible. It remains to show that C is irreducible.

Suppose that C is reducible. By definition there exists a permutation matrix P such that

$$P^T CP = \begin{bmatrix} C_{11} & C_{12} \\ 0 & C_{22} \end{bmatrix}$$

which gives

$$(P^T CP)^k = \begin{bmatrix} C_{11} & C_{12} \\ 0 & C_{22} \end{bmatrix}^k = \begin{bmatrix} B_{11} & B_{12} \\ 0 & B_{22} \end{bmatrix}$$

where B_{11} and B_{22} are quadratic matrices of smaller sizes than C. Any permutation matrix is ortogonal, so we have $P^T = P^{-1}$. This gives that

$$(P^T CP)^k = (P^{-1} CP)^k = P^{-1} C^k P = P^T C^k P$$

By combination we then have that C^k is reducible.

QED

An important result on non-negative matrices is the following theorem.

Theorem 3.5 (Perron - Frobenius)

Let A be a non-negative, quadratic and irreducible matrix. Then

- *A has a positive eigenvalue* λ_1. λ_1 *is the spectral radius of the matrix.*
- *The eigenvector associated with* λ_1 *is positive*
- λ_1 *is a single root in the characteristic equation of A.*

A proof of theorem 3.5 can for example be found in Graham, 1987.

We are now ready to formulate the main result of this section.

Corollary 3.2

Let C be the cost matrix for any symmetric, real instance of TSP, such that $c_{ii} = 0$ and $c_{ij} > 0, i \neq j$. *Then C can be decomposed into a weighted linear combination of n polynomial solvable matrices* T_k *; ie. symmetric product matrices given by the eigenvalues* λ_k *and eigenvectors* $\vec{V}_k$ *of C such that*

1. $C = \sum_{k=1}^{n} \lambda_k T_k$ and $T_k = \left[v_{k_i} v_{k_j} \right]$
2. $\lambda_1 > 0,\ \lambda_1 \geq |\lambda_k|, \forall k$
3. $\vec{V}_1 > \vec{0}$
4. $\sum_{k=1}^{n} \lambda_k = 0$

Proof:

The corollary follows directly from the previous results apart from point *4*. Point *4* follows from (3.2), in that all the elements on the main diagonal are zero and that the set of eigenvectors $\vec{V}_k$ forms an orthonormal basis for the vector space.

QED

Note that the matrix $\lambda_1 T_1$ is a positive matrix, and that all the other matrices in the decomposition can have both negative and positive elements. For all *n* we have at least one negative eigenvalue. The decomposition does *not* represent a linear admissible transformation. As mentioned above, each of the matrix components can be solved to optimality in a polynomial time by theorem 1.10, but of course the optimal cycles for different components need not be the same.

Let φ^* be an optimal permutation for the cost matrix *C*, and φ_k^* denote the minimal Hamiltonian cycle for T_k if the corresponding eigenvalue is positive and the maximal

Hamiltonian cycle for T_k if the corresponding eigenvalue is negative. Corollary 3.2 gives a lower bound for TSP, which can be found in polynomial time.

Corollary 3.3

Upper and lower bounds for an optimal solution in a real symmetric TSP instance are given by the inequalities below:

$$\min_k\{C(\varphi_k^*)\} \geq C(\varphi^*) \geq \sum_{k=1}^{n} \lambda_k T_k(\varphi_k^*)$$

Proof:

The first inequality is trivial. By corollary 3.2 and the definitions of the cycles we have:

$$C(\varphi^*) = \sum_{k=1}^{n} \lambda_k T_k(\varphi^*) \geq \sum_{k=1}^{n} \lambda_k T_k(\varphi_k^*)$$

QED

Note:

Corollary 3.3 can be used as a heuristic for any real symmetric matrix, much in the same way as for example the Clarke and Wright, or the nearest neighbour heuristic in the following sense: The decomposition gives basically *n* different optimal cycles one for each of the components. Each of these applied on the original cost matrix gives an upper bound. In addition we get a lower bound as a bonus.

Let φ be any permutation, and $\psi_k{}^*$ denote the maximal Hamiltionian cycle for T_k if the corresponding eigenvalue is positive and the minimal Hamiltonian cycle for T_k if the corresponding eigenvalue is negative. Further, let λ^+ and λ^- denote the sum of the positive and negative eigenvalues respectively.

Corollary 3.4

Upper bounds for ***any*** *Hamiltonian cycle in a real, symmetric matrix C are*

$$C(\varphi) \leq \sum_{k=1}^{n} \lambda_k T_k(\psi_k^*) \leq 2\lambda^+ \text{ or } C(\varphi) \leq \sqrt{n \sum_{k=1}^{n} \lambda_k^2}$$

Proof:

The first inequality is proven just like the inequality in the previous corollary. The next and poorer bound stems from the fact that the absolute value of the scalar products involved in calculating the maximal or minimal cycles in the symmetric

product matrices all are equal or less than one and that $\lambda^+ = |\lambda^-|$. The last inequality can be proven in the following way:

By lemma 3.1 we have that

$$(C(\varphi))^2 = \left(\sum_{k=1}^{n} \lambda_i < \vec{V}_k, \vec{W}_k >\right)^2 \leq \left(\sum_{k=1}^{n} \lambda_k^2\right)\left(\sum_{k=1}^{n} < V_k, W_k >^2\right) \leq n\left(\sum_{k=1}^{n} \lambda_k^2\right)$$

where the first inequality follows from the well-known Cauchy-Schwarz inequality. The result follows.

QED

By equation (2.3) the original cost matrix can be replaced by any saving matrix. The question is then, whether the saving matrices can be used in the same way as in corollary 3.2 or not? Note however, that a saving matrix is reducible and can contain negative entries, unless the triangle inequality applies. If the triangle inequality holds for the original cost matrix any saving matrix will be non-negative, but it may still happen that the saving matrices contain entries with zero value, outside the row and column no. *d*. On the other hand if the original cost matrix is symmetrical, all saving matrices will be symmetrical as well. Let $S^d(d)$ denote the *(n-1)x(n-1)* matrix obtained from S^d where row and column no. *d* are deleted. The characteristic equation for S^d can then be written as $|S^d - \mu I_n| = \mu|S^d(d) - \mu I_{n-1}| = 0$ which shows that $\mu = 0$ is an eigenvalue for S^d, but we can still not be sure that $S^d(d)$ is irreducible and positive.

If $S^d(d)$ is positive and irreducible then corollary 3.2 applies, and we can find the *n -1* eigenvalues for $S^d(d)$ which together with the last one, $\mu = 0$, can be used to decompose the saving matrix S^d in the same way as for the original cost matrix, ie

$$S^d = \sum_{k=1}^{n} \mu_k R_k \text{ and } R_k = \left[v_{k_i}^d, v_{k_j}^d\right]$$

where $\mu_1 > 0, \mu_1 \geq |\mu_k|, \forall k$, and $\vec{V}_k^d = \left[v_{k_i}^d\right]$ is eigenvector no. *k*, $\vec{V}_1^d > 0$.

Let ψ_k^* denote the maximal Hamiltionian cycle for R_k if the corresponding eigenvalue is positive and the minimal Hamiltonian cycle for R_k if the corresponding eigenvalue is negative. For each *d* Corollary 3.5 gives an upper and lower bound for TSP, which can be found in polynomial time.

Corollary 3.5

If a saving matrix for a real symmetric matrix is positive and irreducible, upper and lower bounds for the optimal solution are given by the inequalities below

$$\min_k \{C(\psi_k^*)\} \geq C(\varphi^*) \geq 2R_d - \sum_{k=1}^{n} \mu_k R_k(\psi_k^*)$$

The lower bound in corollary 3.5 is of the same kind as the lower bound given by (2.19). As in corollary 3.3, corollary 3.7 can be used as a heuristic, yielding lower bounds for the TSP. Combining the two corollaries can give better results than just using one of them.

3.4 Decomposition of m-TSP and the Spectral Theorem

If we have an m-TSP we can transform this slightly more complex situation to a somewhat larger TSP as described in section 1.5 by making *m - 1* copies of the depot, which can be chosen to be node *i = 1*. Let C_m be the cost matrix for this extended TSP, and *C(1)* the matrix *C* where the first row and column are deleted. We then have that

$$C_m = \begin{bmatrix} 0 & D \\ D^T & C(1) \end{bmatrix}_{(n-m+1)x(n-m+1)}$$

where $D = [d_{ij}]_{mx(n-1)} = [c_{1j}]_{mx(n-1)}$ $\forall i = 1,2,\ldots,m$ and $\forall j = 2,3,\ldots,n$

Lemma 3.3

The matrix C_m is irreducible.

Proof:

The matrix *D* consists of *m* identical rows where each entry is the cost from the original depot to each of the other node. Hence *D > 0*, and the transposed matrix will also be non-negative. *C(1)* is a symmetric, non-negative matrix with zeros on the main diagonal. The square of any such matrix will be positive. By ordinary matrix multiplication of partitioned matrices we get

$$C_m^2 = \begin{bmatrix} DD^T & DC(1) \\ C(1)D^T & D^TD + (C(1))^2 \end{bmatrix}$$

It is easily seen that this matrix is positive and hence is irreducible.

QED

Corollary 3.6

The matrix C_m can be decomposed as the matrix C in corollary 3.2 using the corresponding eigenvalues and eigenvectors. The lower bounds given by corollary 3.3 and 3.5 apply in the same way as will the upper bounds in corollary 3.4,

In order to perform the decomposition in corollary 3.8 we have to find the eigenvalues and the corresponding eigenvectors. A natural question to ask is: How do these eigenvalues and eigenvectors relate to the eigenvalues and eigenvectors of the original cost matrix?

To say something about this we need the following theorem, which is called the Cauchy inequalities. A proof can be found in Graham, 1987.

Theorem 3.6

Let A be a symmetrical nxn matrix with eigenvalues $\mu_1 \geq \mu_2 \geq \ldots\ldots\ldots \geq \mu_n$ and B an (n-1)x(n-1) principle sub-matrix of A with eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots\ldots\ldots \geq \lambda_{n-1}$. Then

$$\mu_1 \geq \lambda_1 \geq \mu_2 \geq \lambda_2 \geq \ldots\ldots\ldots \lambda_{n-1} \geq \mu_n$$

This property is called the *interlacing property* of eigenvalues $\mu_1 \geq \mu_2 \geq \ldots\ldots\ldots \geq \mu_n$

Corollary 3.7

Let $\mu_1 \geq \mu_2 \geq \ldots\ldots\ldots \geq \mu_{n+m-1}$ denote the eigenvalues of $C_m = [d_{ij}]$ in corollary 3.8 and $\lambda_1 \geq \lambda_2 \geq \ldots\ldots\ldots \geq \lambda_n$ the eigenvalues of C.

If m = 2 we have that $\mu_1 \geq \lambda_1 \geq \mu_2 \geq \lambda_2 \geq \ldots\ldots\ldots \geq \lambda_n \geq \mu_{n+m-1}$

If m > 2 we have that $\mu_1 \geq \lambda_1$ and $\lambda_n \geq \mu_{n+m-1}$

If $\mu_1 \neq 0$ is an eigenvalue for the matrix C_m the m first co-ordinates for the corresponding eigenvector are equal.

Proof:

The first two statements follow immediately from the interlacing property of eigenvalues since *C is* the principle sub-matrix of C_2 and if $m > 2$, the argument can be extended by taking a principle of a principle etc. of C_m.

Let $\vec{V} = [v_i]$ be the corresponding eigenvector. By definition we then have that $\mu v_i = \sum_{ij=1}^{n+m-1} d_{ij} v_j$ and so $v_i = \frac{1}{\mu} \sum_{ij=1}^{n+m-1} d_{ij} v_j = \frac{1}{\mu} \sum_{ij=1}^{n+m-1} d_{kj} v_j = v_k$. Since the *m* first rows in C_m are identical. QED

3.5 Decomposition of Asymmetric TSP and the Spectral Theorem

In the case of an asymmetric real non-negative square matrix, we cannot any longer be sure that the eigenvalues or the eigenvectors become real numbers but some of the eigenvalues and or eigenvectors will be complex numbers or complex vectors. On the other hand, one advantage of allowing complex numbers as eigenvalues is that all matrices will have eigenvalues since, the characteristic equation can now be solved for any square matrix. In order to pursue this a bit further, we need some more notations and definitions.

In this section, the letter *i* will be exclusively reserved for denoting the *imaginary* unit, that is $i = \sqrt{-1}$. A complex number will be denoted as $z = a+bi$, where a and b are real numbers and sometimes denoted as the real part of z, $Re(z)$ and the imaginary part of z, $Im(z)$, respectively. The (complex) conjugate to a number z is $\overline{z} = a - bi$. The set of complex numbers will be denoted by C. Any matrix with complex entries will be denoted with Latin capital letters other than C.

If A is a matrix with complex entries, the *conjugate transpose* A^* of A is defined as

$$A^* = \overline{A}^t$$

where $\overline{A}$ is a matrix whose entries are the complex conjugate of the corresponding entries in A and $\overline{A}^t$ is the transpose of $\overline{A}$.

A square matrix A with complex entries is called *unitary* iff $A^{-1} = A^*$.

A square matrix A with complex entries is called *Hermitian* iff $A = A^*$

A square matrix A with complex entries is called *skew-Hermitian* iff $A^* = -A$

A square matrix A with complex entries is called *normal* iff $AA^* = A^*A$

Every Hermitian matrix is normal and every unitary matrix is normal.

A square matrix A with complex entries is called *unitarily* diagonalizable iff there exsists a unitary matrix P such that $P^{-1}AP$ is diagonal.

The so-called Euclidean inner product of two vectors in the real case, has to be changed slightly but in an important way to make sense in the complex case.

So, let $\vec{U} = (u_1, u_2, ...u_n)$ and $\vec{V} = (v_1, v_2, ...v_n)$ be two vectors in C^n. Then their Euclidean inner product is defined as:

$$\vec{U}\vec{V} = u_1\overline{v_1} + u_2\overline{v_2} + \ldots\ldots\ldots + u_n\overline{v_n}$$

The basic results for square matrices with complex entries are summed up in the following four theorems:

Theorem 3.7

If A is a nxn matrix with complex entries, then the following are equivalent.

a) *A is unitary*
b) *The row vectors of A form an orthonormal set in C^n with the Euclidean inner product*
c) *The column vectors of A form an orthonormal set in C^n with the Euclidean inner product*

Theorem 3.8

If A is a nxn matrix with complex entries, then the following are equivalent

a) *A is unitarily diagonlizable*
b) *A has an orthonormal set of eigenvectors*
c) *A is normal*

Theorem 3.9

If A is a normal matrix, then the eigenvectors from different eigenspaces are orthogonal

Theorem 3.10

The eigenvalues of a Hermitian matrix are real numbers. The eigenvalues of a skew-Hermitian matrix are pure imaginary numbers.

Note that in any square matrix A, any row vector will be the complex conjugate of the corresponding column vector in the complex conjugate matrix A^*. Now, let P be the complex matrix consisting of the eigenvectors of a unitary matrix, where each eigenvector constitutes a column vector, $\vec{V}_k = (v_{k1}, v_{k2}, \ldots\ldots, v_{kn})$. The complex conjugate matrix $P^* = P^{-1}$ then consists of the row vectors $(\overline{v}_{k1}, \overline{v}_{k2}, \ldots\ldots, \overline{v}_{kn})$. For any unitary matrix A, we then have that $A = PDP^{-1}$ where D is a diagonal matrix with the eigenvalues along the main diagonal.

By usual matrix multiplication we then have for any unitary matrix that

(3.5) $$a_{st} = \sum_{k=1}^{n} \lambda_k v_{ks} \overline{v}_{kt}$$

which is the complex counterpart of (3.2).

In general, the numbers in (3.5) are complex numbers.

Let $\lambda_k = \mu_k + \rho_k i$ where μ_k, ρ_k are real numbers, and $v_{ks} = b_{ks} + d_{ks} i$ and b_{ks} and d_{ks} are real numbers. Then substituting these expressions into the right-hand side of (3.5) we get that

$$\text{(3.6)} \qquad \mathrm{Re}(\lambda_k v_{ks} \bar{v}_{kt}) = \mu_k (b_{ks} b_{kt} + d_{ks} d_{kt}) + \rho_k (b_{ks} d_{kt} - b_{kt} d_{ks})$$

and

$$\text{(3.7)} \qquad \mathrm{Im}(\lambda_k v_{ks} \bar{v}_{kt}) = \mu_k (b_{kt} d_{ks} - b_{ks} d_{kt}) + \rho_k (b_{ks} b_{kt} + d_{ks} d_{kt})$$

For any unitary asymmetric *real* square matrix, the imaginary parts must add up to zero. Hence, for such a matrix we have the following straightforward theorem:

Theorem 3.11

Any unitary asymmetric real square matrix can be decomposed into a sum of 2n symmetric product matrices, plus the sum of n skew-symmetric matrices, formed by n asymmetric product matrices. The vectors forming the product matrices can be found by finding the n eigenvalues and n eigenvectors of the given matrix.

Any normal matrix is unitary. Hence, any normal asymmetric real matrix can be decomposed in the way indicated by theorem 3.11. Such matrices exist, but not every real asymmetric matrix is normal. For normal asymmetric real matrices we have the following straightforward corollary:

Corollary 3.8

Let A be a normal asymmetric real matrix with eigenvalues and eigenvectors defined as above. Then there exists 4n real vectors $\overrightarrow{b_k}, \overrightarrow{d_k}, \overrightarrow{\mu_k}$ and $\overrightarrow{\rho_k}$ *such that*

$$A = \sum_k \mu_k B_k + \sum_k \mu_k D_k + \sum_k \rho_k E_k$$

where B_k and D_k *are polynomial solvable symmetric product matrices* $[b_{k_s} b_{k_t}]$ and $[d_{k_s} d_{k_t}]$, *respectively, and* E_k *are skew-symmetric matrices* $[b_{k_s} d_{k_t} - b_{k_t} d_{k_s}]$.

Circulant matrices

Such instances of TSP are "almost" polynomial solvable, since the shortest Hamiltonian path can be found very easily. However, it is still an open question whether the TSP for circulants is polynomial solvable or not.

Note that for a matrix to be normal, we must have that $AA^*=A^*A$. This is equivalent with

$$\forall p,q; \sum_{k=1}^{n} a_{pk} a_{kq}^{*} = \sum_{k=1}^{n} a_{pk}^{*} a_{kq}$$

where a_{pq}^{*} are the entries in A^{*}. Due to the special structure of circulant matrices it is easily shown that this class of matrices is normal. Hence, circulant matrices can be decomposed according to corollary 3.10 above. Moreover, circulant matrices of the same size have identical eigenvectors whatever the entries may be. These eigenvectors are given by the so-called Fourier-matrix. The entries of the Fourier matrix F^{*} of size n is given by:

$$\frac{1}{\sqrt{n}} w^{(i-1)(j-1)}$$

where w can be taken as any primitive *n-th* root of unity, or if one prefers:

$$w = e^{\frac{2\pi i}{n}} = \cos\frac{2\pi}{n} + i\sin\frac{2\pi}{n}$$

A circulant matrix C is characterised by n different entries $(c_1, c_2, c_3, ..., c_n)$ where the first is on the main diagonal, the next on the first stripe and so on. The eigenvalues of the matrix is then given by

$$\lambda_j = c_1 + c_2 w^{j-1} + c_3 w^{2(j-1)} + ... + c_n w^{(n-1)(j-1)}$$

For details and proofs, see for example Davis, 1979.

Consequently, circulant matrices are in many ways easy to describe, and due to their special structures explicit formulas for their eigenvalues and eigenvectors exist. However, even being given these neat properties, it seems as if the spectral theorem does not solve the open question about the polynomial solvability of circulants.

3.6 Hermitian Matrices and Asymmetric Real Matrices

As noted above, not every asymmetric real matrix is normal. So for a non-normal asymmetric real matrix, we cannot be sure that it is unitarily diagonalizable. On the other hand, any asymmetric matrix A can be decomposed into a symmetric matrix D and a skew-symmetric matrix E in the following simple way:

$$A = D + E = \left[\frac{a_{st} + a_{ts}}{2}\right] + \left[\frac{a_{st} - a_{ts}}{2}\right]$$

Hence, to each asymmetric real matrix A, we can associate a Hermitian matrix H_A, where $h_{st} = d_{st} + e_{st} i$. On the other hand, given any Hermitian matrix, we can from

this associate a real asymmetric matrix, just by adding the real and imaginary parts of the complex entries in the Hermitian matrix.

Now, let φ be a Hamiltonian cycle in an asymmetric TSP instance with real cost elements. Then naturally the cost of this cycle, $A(\varphi)$, will be a real number, whilst the "cost", $H_A(\varphi)$, in the corresponding Hermitian matrix may be a complex number. So, to measure the cost of the cycle with the help of a Hermitian matrix cannot be done directly, but it is easily observed that $A(\varphi) = \mathrm{Re}(H_A(\varphi)) + \mathrm{Im}(H_A(\varphi))$. Hence, to find the minimal cycle in *A*, will be the same as finding the cycle where the sum of the real and imaginary parts of the complex numbers in the corresponding Hermitian matrix is as small as possible.

Now, all eigenvalues for Hermitian matrices are real numbers. This means that all ρ_k in (3.6) and (3.7) above are zero. Hence the sum of the real and imaginary parts of a Hamiltonian cycle in a Hermitian matrix will be the sum of elements of the following type:

$$[\sum_k \mu_k (b_{k_s} b_{k_t} + d_{k_s} d_{k_t}) + (b_{k_t} d_{k_s} - b_{k_s} d_{k_t})]$$

This gives the following theorem.

Theorem 3.12

Let A be an asymmetric real matrix and let the corresponding Hermitian matrix H_A have eigenvalues and eigenvectors as defined above. Then there exists 3n real vectors $\overrightarrow{b_k}, \overrightarrow{d_k}$ and $\overrightarrow{\mu_k}$ such that

$$A = \sum_k \mu_k B_k + \sum_k \mu_k D_k + \sum_k \mu_k E_k$$

where B_k and D_k are polynomial solvable symmetric product matrices $[\, b_{k_s} b_{k_t} \,]$ and $[\, d_{k_s} d_{k_t} \,]$ respectively, and E_k are skew-symmetric matrices $[\, b_{k_s} d_{k_t} - b_{k_t} d_{k_s} \,]$.

In a similar fashion, we can assign to each asymmetric matrix *A* a skew-symmetric Hermitian matrix H_{SA} by defining the entries in this matrix as $h_{st} = e_{st} + d_{st} i$. For skew-symmetric Hermitian matrices the eigenvalues will have pure imaginary values. Hence, the μ_k in (3.6) and (3.7) will become zero, leaving a decomposition of the same kind as in theorem 3.12.

Final comment

Compared with real symmetric matrices, decompositions of real asymmetric normal matrices with the help of the spectral theorem are more complicated, and indicate that there may exist a significant difference between solving STSP and normal ATSP. The same is the case when decomposition is made with the help of a Hermitian matrix. The former can be decomposed into a sum of polynomial solvable matrices, while the latter one seems to need a more complex decomposition involving matrices of a kind that are known to be NP-hard. Viewed from the perspective of the spectral theorem,

then the ATSP appears to be more difficult than the STSP case. It has been noted by several authors, see for example Burkhardt 1979, that the numerical behaviour of symmetric TSP is completely different from that of an asymmetric one. Some of these differences may be explained by the differences of decompositions exposed in this chapter.

Chapter 4

The different decompositions in the previous chapters are of course only some of infinitely many different possibilities. In this chapter other decomposition schemes for ATSP will be explored, mainly using a constant TSP matrix, a symmetric matrix and a residual asymmetric matrix. In the second section an instance of TSP is decomposed into a constant-TSP matrix, a symmetric matrix and a residual asymmetric matrix. In addition, a new situation is considered where a sum of two Hamiltonian cycles is solved simultaneously, the cycles being reversed of each other. The third and last section contains decompositions of a matrix into a constant-TSP, a symmetric matrix, and a skew-symmetric matrix, where the last one can be taken to be minimal in a certain sense. However, we start this chapter with a sub-class of ATSP where the decomposition is very simple.

4.1 Hamiltonian Symmetric Travelling Salesman Problems

Usually, an asymmetric cost matrix will give at least some Hamiltonian cycles with different costs when the sequence of nodes is taken in reversed order. However, this is not always the case. For instance, in a constant-TSP, which in general will be asymmetric, every Hamiltonian cycle will have the same cost.

A natural question to ask is: do asymmetric cost matrices exist such that they do not yield a constant-TSP, but where every Hamiltonian cycle and its reverse have the same cost? The answer to this question is yes.

Such a subset of ATSP is readily described by any matrix C such that:

(4.1) $$c_{ij} = a_i + b_j + d_{ij}$$

where $D = [d_{ij}]_{nxn}$ is a symmetric matrix and $K = [a_i + b_j]_{nxn}$ is a constant-TSP asymmetric matrix. Note that the transformation in (4.1) is a linear admissible transformation. It is evident that the cost of any Hamiltonian cycle in every matrix C of this kind will consist of a constant-TSP part where, all asymmetric aspects of the matrix C are cancelled out. The symmetric part, which of course can give different values for different Hamiltonian cycles, will nevertheless give equal costs for a Hamiltonian cycle and its reverse.

We will as before use lower case Greek letters for denoting cyclic permutations reflecting a Hamiltonian cycle in the graph. For instance:

$$\vec{\varphi} = i_1 - i_2 - i_3 - \ldots\ldots - i_n - i_1$$

is a Hamiltonian cycle. The reverse will be denoted by:

$$\overleftarrow{\varphi} = i_1 - i_n - i_{n-1} - \ldots\ldots\ldots i_2 - i_1$$

A natural question to ask is whether there exist other classes than the one given by (4.1) with the same property? The answer to this question is negative as lemma 4.1 demonstrates.

Lemma 4.1

Let C be a non-negative nxn matrix. We then have:

$\forall \varphi, C(\vec{\varphi}) = C(\overleftarrow{\varphi}) \Leftrightarrow \exists K, D \geq 0, D = \left[d_{ij}\right]_{nxn}$ symmetric ; $K = \left[a_i + b_j\right]_{nxn}$ and $c_{ij} = a_i + b_j + d_{ij}$

Proof:

If K and D are as stated, then it is incidentally true by the comment at the beginning of this section, that any Hamiltonian cycle and its reverse have the same cost.

In order to prove the second half of the lemma, we first solve the following LP-problem:

$$(4.2) \qquad \max \sum_{i=1}^{n} a_i + \sum_{i=1}^{n} b_j + \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}$$

ST

$$(4.3) \qquad a_i + b_j + d_{ij} \leq c_{ij} \quad \forall i, j; i \neq j$$

$$(4.4) \qquad d_{ij} = d_{ji} \quad \forall i, j$$

all variables non - negative

The solution to this LP-problem will create a matrix K and a symmetric matrix D, both non-negative, such that :

$$a_i + b_j + d_{ij} \leq c_{ij}$$

for all i and j.

Let P be the set of edges such that:

$$P = \left\{(i,j) \middle| c_{ij} > a_i + b_j + d_{ij}\right\}$$

and suppose that $P \neq \varnothing$ and let $(s,t) \in P$. Then $(t,s) \notin P$ and hence $c_{ts} = a_t + b_s + d_{ts}$, because if $c_{ts} > a_t + b_s + d_{ts}$, we could have chosen a larger d_{st} in the solution of the LP-model. $\forall (i,j) \in P$, we then have:

$$c_{ij} - c_{ji} > (a_i + b_j + d_{ij}) - c_{ji} = (a_i - a_j) + (b_j - b_i)$$

For every pair *(i,j)* not in *P* we will have equality in the above inequality.

Let $\vec{\varphi}$ be any cyclic permutation that contains at least one pair from *P*. We then have:

$$C(\vec{\varphi}) - C(\overleftarrow{\varphi}) = \sum_{s=1}^{n} c_{s,s(\varphi)} - \sum_{s=1}^{n} c_{s(\varphi),s} > \sum_{s=1}^{n} \left[(a_s - a_{s(\varphi)}) + (b_{s(\varphi)} - b_s) \right] = 0$$

Which gives that $C(\vec{\varphi}) > C(\overleftarrow{\varphi})$. Hence we have $P \neq \varnothing \Rightarrow \exists \vec{\varphi}; C(\vec{\varphi}) \neq C(\overleftarrow{\varphi})$ or equivalent $\forall \vec{\varphi}, C(\vec{\varphi}) = C(\overleftarrow{\varphi}) \Rightarrow P = \varnothing$, that is, there is no slack in the solution of the LP-problem above and every inequality holds with equality.

QED

Lemma 4.1 justifies the following definition:

Definition 4.1

An asymmetric non-negative matrix, where the elements can be formed as $c_{ij} = a_i + b_j + d_{ij}, d_{ij}$ symmetric and all numbers are non-negative, is called ***Hamiltonian symmetrical.***

In a different context, namely for the Chinese postmen problem for *k* different postmen, Pearne defines these cycles as (just) symmetrical, see Pearne, 1994. However, to differentiate between a symmetric matrix as such, and an asymmetric matrix where each Hamiltonian cycle and its reverse have the same cost, definition 4.1 seems to be more appropriate.

It is easy to check whether a given asymmetric non-negative matrix is Hamiltonian symmetrical or not. One just solves the LP-model in the proof of lemma 4.1. If the optimal solution does not give any positive slack in the restrictions, then the matrix is of the said kind. Other criteria are given in the theorem 4.2 below, but first it is convenient to have a look at the set of Hamiltonian symmetric matrices as such.

It is easily seen that the set of all Hamiltonian symmetric matrices of a given size forms a vector space. The theorem below gives the dimensions of these vector spaces.

Theorem 4.1

The dimension of the vector space consisting of all the nxn Hamilton symmetrical matrices is $2n-1+\frac{n(n-1)}{2}-1=\frac{(n-1)(n+4)}{2}$.

Proof:

Let R_i be the *nxn* matrix which has 1 at each element in the *i-th* row and zeros elsewhere, K_j is the *nxn* matrix with 1 in the *j-th* column and zeros elsewhere, and finally, $M_{ij}, i \neq j$ the *nxn* matrix with 1 in the positions *(i,j)* and *(j,i)* and zeros elsewhere.

As observed by Berenguer, 1979, any *2n-1* of the *2n* matrices R_i and K_j forms a linear independent basis for the vector space of all constant-TSP matrices of the same size. Similarly any *n(n-1):2 -1* of the *n(n-1):2* matrices M_{ij} forms a linear independent basis for the vector space consisting of all symmetric matrices of the given size. It is easily checked that the combination of these sets of vectors forms a linear basis for the stated vector space.

QED

Corollary 4.1

Let C be Hamiltonian symmetric and let S be a set of sub-cycles in the underlying graph which forms a feasible solution to a m-TSP or VRP problem. Then the cost of this solution will be the same as for the solution formed by the reversed sequence of nodes.

Proof:

This follows directly from the fact that the property of a constant-TSP matrix holds equally well for a set of sub-cycles as for a complete Hamiltonian cycle.

QED

Note:

Given a Hamiltonian symmetric matrix one can solve the TSP with the help of model 2, instead of using model 1, see chapter 1. Model 1 usually calls for more facets or sub-tour eliminating restrictions than model 2.

Corollary 4.2

Any Hamiltonian symmetric matrix can be decomposed into a constant-TSP matrix and a sum of symmetric product matrices.

Proof:

The result follows directly from the comments above, and the decomposition for symmetric matrices done by the spectral theorem in the previous chapter.

QED

Hence, this sub-class of asymmetric TSP instances is as easy or difficult to solve as symmetric instances.

As a last point in this section, the following theorem shows that any Hamiltonian symmetric matrix can be characterised in different ways. The theorem gives several equivalent ways of identifying such matrices.

Theorem 4.2

Let C be any nxn matrix. Then the following statements are equivalent:

(a) *C is Hamiltonian symmetrical*

(b) *C = K + D, where K is a constant-TSP matrix and D is symmetrical*

(c) $S^k(C)$ *is symmetrical for any node k*

(d) $$c_{ij} - c_{ji} = \frac{1}{n}\left(\left(R_i(C) - K_i(C)\right) - \left(R_j(C) - K_j(C)\right)\right) \quad \forall i, j; i \neq j$$

Proof:

The equivalence between *(a)* and *(b)* is established from the results above. Suppose that *(b)* holds. Note that every symmetric matrix has symmetrical saving matrices. Then we have:

$$s_{ij}^k(C) = c_{ik} + c_{kj} - c_{ij} = (a_i + b_k + d_{ik}) + (a_k + b_j + d_{kj}) - (a_i + b_j - d_{ij}) =$$
$$a_k + b_k + (d_{ik} + d_{kj} - d_{ij}) = a_k + b_k + s_{ij}^k(D) = a_k + b_k + s_{ji}^k(D) = s_{ji}^k(C)$$

which gives *(c)*. Then let *(c)* be given. Every Hamiltonian cycle for any matrix can be solved either directly with the original matrix C - here asymmetric - or by finding a corresponding Hamiltonian cycle in any of its saving matrices. The values of the corresponding cycles are given by equation (2.4). Clearly every Hamiltonian cycle in $S^k(C)$ has the necessary property. In the next section *(d)* will be dealt with.

QED

4.2 Optimal Pairs of Hamiltonian Cycles

A salesman travelling in a symmetric matrix need not concern himself about the costs incurred, regardless of whether he travels in one direction or the other along a certain cycle. This will in general not be the case if the matrix is genuinely asymmetric, but not Hamiltonian symmetric. However, a slightly different situation will occur if the salesman for some reason wants to travel a certain cycle in one direction at one time, and every second time in the opposite direction. Even if one of the two sequences is optimal, the sum of the costs of the two cycles does not need to be the smallest one.

It turns out to be possible to find "an optimal pair" of Hamiltonian cycles for asymmetric matrices with the help of an appropriate symmetric matrix. It will be convenient to introduce the following concepts:

Definition 4.3

Let C be any (genuine) asymmetric matrix and let φ be any Hamiltonian cycle. With an ***asymmetric pair*** *φ we will understand the pair of Hamiltonian cycles $(\vec{\varphi}, \overleftarrow{\varphi})$. With every asymmetric pair we will associate the* ***asymmetric cost*** *$A(\vec{\varphi}) + A(\overleftarrow{\varphi})$ for the asymmetric pair. With the* ***optimal*** *asymmetric pair we will understand the asymmetric pair with the smallest asymmetric cost*

We can now formulate the following theorem:

Theorem 4.4

For any asymmetric matrix A, the optimal pair can be found by solving TSP in an appropriate symmetric matrix.

Proof:

In subsection 3.1.6 it was noted that any asymmetric matrix could be decomposed respectively into a symmetric matrix D and a skew-symmetric E matrix such that $A = D + E$. Let φ be any Hamiltonian cycle.

Then $A(\varphi) = D(\varphi) + E(\varphi)$ and $A(\overleftarrow{\varphi}) = D(\varphi) - E(\varphi)$. Added together, these two expressions yield that $A(\varphi) + A(\overleftarrow{\varphi}) = 2D(\varphi)$ and the result follows.

QED

Corollary 4.6

Let φ^* and ψ^* be the optimal Hamiltonian cycles in A and D respectively. We then have following string of inequalities:

$$A(\varphi^*) \le \min\left\{A(\psi^*), A(\overleftarrow{\psi^*})\right\} \le \frac{A(\psi^*) + A(\overleftarrow{\psi^*})}{2} \le A(\overleftarrow{\varphi^*}) \le \max\left\{A(\psi^*), A(\overleftarrow{\psi^*})\right\}$$

Proof:

The two first inequalities are trivial. Since ψ^* denotes the Hamiltonian cycle for the optimal pair, we have that $A(\varphi^*) + A(\overleftarrow{\varphi^*}) \geq A(\psi^*) + A(\overleftarrow{\psi^*})$ Further, we have that $A(\varphi^*) \leq A(\overleftarrow{\varphi^*})$ which means that $2A(\overleftarrow{\varphi^*}) \geq A(\psi^*) + A(\overleftarrow{\psi^*})$ which then gives the two last inequalities.

QED

Corollary 4.7

Let φ^* and ψ^* be the optimal Hamiltonian cycles in A and D respectively. Then we have that

$$2D(\psi^*) - \max\left\{A(\psi^*), A(\overleftarrow{\psi^*})\right\} \leq A(\varphi^*)$$

Proof:

From the proof of theorem 4.4 and the definitions of the cycles we have that

$$A(\varphi^*) + A(\overleftarrow{\varphi^*}) = 2D(\varphi^*) \geq 2D(\psi^*)$$

Hence,

$$A(\varphi^*) \geq 2D(\psi^*) - A(\overleftarrow{\varphi^*})$$

By corollary 4.6 we get that

$$2D(\psi^*) - A(\overleftarrow{\varphi^*}) \geq 2D(\psi^*) - \max\left\{A(\psi^*), A(\overleftarrow{\psi^*})\right\}$$

and the result follows.

QED

Corollary 4.6 gives lower and upper bounds for the reversed Hamiltonian cycle of the optimal solution of TSP in *A*, and we can find these bounds by solving a symmetric TSP in an appropriate cost matrix. Note also that the lower bounds described in chapter 2 for symmetric matrices combined with corresponding saving matrices can of course be applied on the symmetric matrix used for finding the optimal pair, yielding lower bounds for the cost of the optimal pair as such. Corollary 4.7 gives a lower bound for the asymmetric matrix calculated by the optimal solution in a symmetric matrix.

Corollary 4.8

Let ψ^* be the optimal solution the symmetric matrix in the decomposition. If $A(\psi^*) = A(\overleftarrow{\psi^*})$ then ψ^* is the optimal solution to the asymmetric matrix too.

Proof:

For any cycle we have that $A(\varphi^*) + A(\overleftarrow{\varphi^*}) = 2D(\varphi^*)$. The right-hand side of this equation will obtain its smallest value for the cycle ψ^*. By assumption of the corollary it follows that $A(\psi^*) = D(\psi^*)$, and hence ψ^* will be the optimal cycle for A as well.

QED

Corollary 4.8 states in a way that if A is a sort of restricted Hamiltonian symmetric matrix, that is: If the cost of the *optimal* Hamiltonian cycle and its reverse are the same, this optimal solution can be found with the help of a symmetric matrix. Another way of illustrating this is to sort the Hamiltonian cycles into two sets. One set consists of all Hamiltonian cycles where the costs of this cycle and its reverse are equal, and the second set of cycles where these costs are different. It may happen that for certain instances of TSP that the first set is a rather large set and the second fairly small, and that the optimal solution belongs to the first set.

4.3 An Algorithm for the Asymmetric Travelling Salesman Problem

In this section we will present an algorithm for the asymmetric TSP which tells the user how close the solutions are to the optimum, and in many cases one can prove the optimality of the best solution found as part of the procedure. The heuristic basically consists of finding the optimal solutions to a related symmetric problem, or near optimal solutions to the same symmetric problem.

Let C be an *nxn* asymmetric non-negative matrix and $a_i, b_j, d_{ij}, e_{ij}, d_{ij} = d_{ji}$ all be non-negative numbers.

The heuristic

Step 0: Solve the LP-model:

$$(4.5) \qquad \max \sum_{i=1}^{n} a_i + \sum_{i=1}^{n} b_j + \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}$$

ST

$$(4.6) \qquad a_i + b_j + d_{ij} + e_{ij} = c_{ij} \quad \forall i, j; i \neq j$$

$$(4.7) \qquad d_{ij} = d_{ji} \quad \forall i, j$$

all variables non - negative

Step1: Find an optimal solution φ^* to the symmetric TSP in $D = [d_{ij}]_{nxn}$

a) If C turns out to be Hamiltonian symmetric, that is $E = [e_{ij}]_{nxn} = 0$,

then stop and calculate the optimal value $C(\varphi^*) = \sum_{i=1}^{n} a_i + \sum_{j=1}^{n} b_j + D(\varphi^*)$.

b) If $E = [e_{ij}]_{nxn} \geq 0$ then denote the value $C(\varphi^*) = C(\varphi_1)$ as the first lower bound LB_1. Further calculate

$$C(\varphi^*) = \sum_{i=1}^{n} a_i + \sum_{j=1}^{n} b_j + D(\varphi^*) + \min\left\{E(\vec{\varphi_1}), E(\overleftarrow{\varphi_1})\right\}$$

and denote this value as the first upper bound UB_1. Go to step 2.

Step k:

Solve model 2 in chapter 1 with the added constraint (4.8).

$$(4.8) \qquad \varphi_k \notin \{\varphi_1, \varphi_2, \ldots\ldots \varphi_{k-1}\}$$

Denote the new optimal cyclic permutation as φ_k where d_e denotes the cost of edge e in the symmetric matrix D. (4.8) affirms that we shall exclude from our new solution all previously found Hamiltonian cycles.

Calculate $(K + D)(\varphi_k) = \sum_{i=1}^{n} a_i + \sum_{j=1}^{n} b_j + D(\varphi_k)$ and let

$LB_k = \max\{LB_{k-1}, (K + D)(\varphi_k)\}$

Calculate $C(\varphi_k) = \sum_{i=1}^{n} a_i + \sum_{j=1}^{n} b_j + D(\varphi_k) + \min\left\{E(\vec{\varphi_k}), E(\overleftarrow{\varphi_k})\right\}$ and let

$UB_k = \min\{UB_{k-1}, C(\varphi_k)\}$

If the current upper and lower bounds are equal, then stop. If not, go to *step k+1*.

It is always possible to find a solution to the model in *step 0*.

It is obvious that this procedure sooner or later will terminate inasmuch as the lower bounds are increasing, and that the best upper bound will decrease, or at least be equal to the previous one.

Note:

That the claimed upper bounds really are upper bounds, follows from the fact that they simply are feasible solutions - that is Hamiltonian cycles - in the asymmetric matrix *C*.

The first lower bound is clearly a lower bound for any Hamiltonian cycle in *C*. This follows from the fact that all elements in this matrix are equal or less than the corresponding elements in *C*. Suppose now that there exists a Hamiltonian cycle constructed by the heuristic, with a cost in *K+D* that is larger than the optimal value in *C*. The permutation that then gave the optimal value in *C* however, must have been encountered ahead of this step and we would already have reached the optimum.

The procedure will produce two Hamiltonian cycles in each step. For each step one or both of these may or may not be better than the best of the previous cycles. Each new step in the procedure may or may not produce a larger lower bound for the ATSP. Whether one gets a new and better lower bound in step *k* or not, will depend on whether there exists another Hamiltonian cycle in *D* with the same cost or not.

The technique in *step* [illegible] used by Jeromin and Körner, 1985, in a simpler version and in another context. They [illegible] use the symmetric part nor the residual asymmetric matrix *E*, but only the constant-TSP part. Their main aim is to improve on the worst-case ratio for heuristics by reformulating the original cost matrix, by ensuring that the new matrix observes both the triangle inequality and being non-negative. Note also that the value of the constant-TSP found in step 0, is exactly the lower bound found by Lagrangean relaxation in model 1 by relaxing both the restrictions.

The procedure above involves solving a symmetric TSP. In many cases this can be a hard task. An alternative is to replace the symmetric matrix *D* by a matrix D_{Pol}, that is by a polynomial solvable matrix of some kind, but not necessarily symmetric. We then have an approximation situation like the one suggested by Burkhard. Whether it will be a good or bad policy to take the constant-TSP into the calculations, will probably depend both on the original matrix and the chosen polynomial solvable class.

Closely related to an upper bound found by a heuristic is the so-called worst case ratio. The best known worst case ratio known so far is that for Christofides' heuristic based on minimal spanning trees and matching technique. This ratio is 1.5, and ensures that any Hamiltonian cycle generated by this heuristic never exceeds 50% of the cost of the optimal solution.

The following lemma gives the worst case ratio for the present procedure, which hardly can be regarded as a heuristic in the traditional sense, since it is based on solving a symmetric TSP to optimum. Nonetheless, based on the more or less well-founded assumption that solving STSP is easier than solving ATSP, there may be some virtue in applying the procedure.

Lemma 4.2

The worst case ratio for the present procedure is 2.

Proof:

Step 0 in the heuristics gives values for the different elements in all the three matrices *K*, *D* and *E*, such that the sum of the elements in *K* and *D* is maximal.

Suppose that $a_i + b_j + d_{ij} < e_{ij}$ for some i and j. Then, the sum cannot be maximal since it would then have been possible to increase either a_i, b_j *or* d_{ij}. Hence, for all *i* and *j*, we have that $a_i + b_j + d_{ij} \geq e_{ij}$. We then have the following calculation for the worst case ratio;

$$\frac{C(\varphi_k)}{C(\psi^*)} \leq \frac{K(\varphi_k)+D(\varphi_k)+E(\varphi_k)}{K(\varphi_k)+D(\varphi_k)} = 1+\frac{E(\varphi_k)}{K(\varphi_k)+D(\varphi_k)} \leq 1+1=2$$

QED

Note:

If one can prove - and I have been unable to do so - that $a_i + b_j \geq e_{ij}$ and $d_{ij} \geq e_{ij}$ for all *i* and *j* in general, or if one has special structured matrices where this is the case, the worst case ratio in lemma 3.2 can be replaced by 1.5. Further, if one has the following three conditions $d_{ij} \geq e_{ij}$, $a_i \geq e_{ij}$ and $b_j \geq e_{ij}$ for all *i* and *j*, the worst case ratio becomes 1.33. This will not hold in general however, as shown by the example in the appendix. It may nevertheless be possible that these conditions hold in special structured matrices.

The above procedure relies on dividing the original asymmetric matrix into three different matrices. For the heuristic to behave well, the numbers in the *E* matrix should not be too large, or rather the gap between the first found lower and upper bounds should not be too large, if one wants to find the optimal solution of the ATSP. Another way of looking at this is to say that the heuristic will behave better, if the variation of the elements in *E* is small. As a consequence, the costs of different cycles measured in *K*+*D* will contain the main part of the costs of the cycles in *C*. The variations of the costs for the same cycles measured in *E*, will not vary so much, due to the relatively small variations in the elements in *E*.

However, as mentioned above, a matrix can be divided into two or more matrices in infinitely many ways. The point here being that the chosen way consists of three different matrices where the difficulties concerning TSP are increasing.

Another way of dividing the original ATSP matrix is to take away as much as possible of the symmetric part of the asymmetric matrix, that is, we claim K in step 0 to be zero or more straightforward, construct the symmetric matrix

$$d_{ij} = \min_{i,j}\{c_{ij}, c_{ji}\}$$

The procedure can be performed using this symmetric matrix instead of dividing the original asymmetric matrix into three parts, and hence not "taking away as much of the asymmetry" as is possible. However, this simplification often has two effects: The generated permutations do not yield as good solutions as fast as in the first case, and it can be very hard to prove optimality.

As mentioned above, the numbers in the residual matrix E should not be too large in order to ensure that the procedure works well and fast. A similar effect can be obtained if the values of the elements of E are fairly constant. Then Hamiltonian cycles in E will have costs that do not differ very much. Hence, the variation of the values of the Hamiltonian cycles in the original cost matrix C, will mostly occur as different costs in the corresponding cycles in D. Hence, the optimal cycle in D, or a near optimal one, will be close to the optimal one in C. Stated in another way, C is close to being a Hamiltonian symmetric matrix.

One way of obtaining smaller numbers in the matrix E in the procedure is to perform an iteration process. The consequences of this process will be that the values of the symmetric matrix D may become larger, and the values of the constant-TSP matrix K may change and in many cases make the corresponding constant-TSP smaller.

The iteration process can be performed as follows:

Solve the LP-model in step 0 in the heuristic. If the $E = [0]$, then we stop. If not, add a new restriction to the model, forcing the values of the elements in the matrix E to become less, that is any e should be less than some number less than the largest e found in the previous try. Solving the LP-model once more we can either obtain a new solutions with new matrices K, D and E, but where the elements of E are smaller than in the previous case, or we may have a model without any feasible solution. In the first case, we choose a more restricted upper bound for the elements in E and solve the model again. In the second case, we relax the upper bound on the elements of E in order to obtain a feasible solution.

It is fairly easy to find a matrix E with minimal elements in this sense. The consequences can be that applying this triple of matrices to the procedure, one may obtain better results faster. In a sense, we have by the iteration process obtained a matrix E, where the variation of the elements is minimal.

The same procedure can of course also be applied if we choose $D = D_{pol}$ for some polynomial solvable class.

However, there are other ways to divide the matrix C into different parts, relaxing the condition that the residual asymmetric matrix has to be non-negative. It will still be important to have a variation as small as possible among the elements of E.

One way of obtaining this is to solve the following problem:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} (e_{ij})^2 \tag{4.9}$$

ST

$$c_{ij} = e_{ij} + d_{ij} \quad \forall i, j; i \neq j \tag{4.10}$$

$$d_{ij} = d_{ji} \quad \forall i, j; i \neq j \tag{4.11}$$

Since the objective is convex, we obtain the optimal solution by the first order conditions. After first substituting (4.10) into the objective and setting the first derivatives equal to zero, one gets:

$$d_{ij} = \frac{c_{ij} + c_{ji}}{2} \quad \forall i, j; i \neq j \tag{4.12}$$

which then by (4.10) gives:

$$e_{ij} = \frac{c_{ij} - c_{ji}}{2} \quad \forall i, j; i \neq j \tag{4.13}$$

Since the matrix C is supposed to be non-negative, D will be non-negative as well. Note that this decomposition is exactly equal to the one used in constructing the Hermitian matrix in sub-section 3.6 and the way we decomposed an asymmetric matrix in order to find the optimal pair in section 4.2. Hence, this decomposition has the property that the variation of the elements in the residual matrix is minimal in the sense above. This gives the following theorem.

Theorem 4.2

Any genuinely asymmetric matrix C can be divided into two matrices C = D + E, where D is non-negative and symmetrical, and where E is skew-symmetrical and the variation of its elements is minimal.

Theorem 4.2 has the explicit premise that the original matrix shall be split into **two** different matrices - one symmetrical and one asymmetrical - giving a result which minimises the variation of the elements of the asymmetrical part of the decomposition. In this case, we do not utilise the possibility that some of the asymmetry of the original cost matrix C can be taken care of by a constant-TSP matrix. This is done in the next theorem.

Theorem 4.3

Any asymmetric matrix C can be divided into three matrices C = K + D + E, where K is a constant-TSP matrix, D is symmetrical and non-negative and E is skew-symmetrical and the variation of its elements is minimal.

The elements of the three different matrices are given by:

i) $K = [a_i + b_j]$ *where the b's can be chosen freely, but such that*

$$b_i + b_j \le \frac{1}{2n}\left(\left(R_i(C) - K_i(C)\right) + \left(R_j(C) - K_j(C)\right)\right) \text{and}$$

$$a_i = b_i + \frac{1}{n}\left(R_i(C) - K_i(C)\right)$$

ii) $$d_{ij} = \frac{c_{ij} + c_{ji}}{2} + \frac{1}{2n}\left[\left(R_i(C) - K_i(C)\right) + \left(R_j(C) - K_j(C)\right)\right] - \left(b_i + b_j\right) \quad \forall i, j; i \neq j$$

iii) $$e_{ij} = \frac{c_{ij} - c_{ji}}{2} - \frac{1}{2n}\left[\left(R_i(C) - K_i(C)\right) - \left(R_j(C) - K_j(C)\right)\right] \quad \forall i, j; i \neq j$$

Proof:

The theorem is derived from solving the following convex minimising problem:

$$(4.14) \qquad \min \sum_{i=1}^{n} \sum_{j=1}^{n} \left(e_{ij}\right)^2$$

ST

$$(4.15) \qquad c_{ij} = a_i + b_j + e_{ij} + d_{ij} \quad \forall i, j; i \neq j$$

$$(4.11) \qquad d_{ij} = d_{ji} \quad \forall i, j; i \neq j$$

Naming the objective *F*, substituting (4.15) into the objective and making the partial derivatives gives:

$$\frac{\partial F}{d_{ij}} = -2(c_{ij} - a_i - b_j - d_{ij}) - 2(c_{ji} - a_j - b_i - d_{ij}) \; \forall i, j; i \neq j \text{ and}$$

$$\frac{\partial F}{a_i} = (-2)\sum_{\substack{j=1 \\ j \neq i}}^{n} (c_{ij} - a_i - b_j - d_{ij}) \quad \forall i$$

$$\frac{\partial F}{b_j} = (-2)\sum_{\substack{i=1 \\ i \neq j}}^{n} (c_{ij} - a_i - b_j - d_{ij}) \quad \forall j$$

Setting the first of the derivatives equal to zero gives

(4.17) $$d_{ij} = \frac{c_{ij} + c_{ji}}{2} - \frac{a_i + a_j}{2} - \frac{b_i + b_j}{2} \quad \forall i, j; i \neq j$$

Summing (4.17) over j we get

(4.18) $$R_i(D) = \frac{1}{2}R_i(C) + \frac{1}{2}K_i(C) - \frac{n-1}{2}a_i - \frac{1}{2}\sum_{\substack{j=1 \\ j\neq i}}^{n} a_j - \frac{n-1}{2}b_i - \frac{1}{2}\sum_{\substack{j=1 \\ j\neq i}}^{n} b_j =$$
$$\frac{1}{2}\left[R_i(C) + K_i(C) - (n-2)a_i - \sum_{j=1}^{n} a_j - (n-2)b_i - \sum_{j=1}^{n} b_j\right]$$

The derivative with respect to a gives after summing over j, j different from i

(4.19) $$R_i(C) - (n-1)a_i - \sum_{\substack{j=1 \\ j\neq i}}^{n} b_j - R_i(D) = 0$$

Substituting (4.18) into (4.19) gives:

$$R_i(C) - K_i(C) - na_i + nb_i + \sum_{j=1}^{n} a_j - \sum_{j=1}^{n} b_j = 0$$

Since we have a certain degree of freedom, we can choose to let $\sum_{j=1}^{n} a_j = \sum_{j=1}^{n} b_j$, which then gives:

(4.20) $$a_i = b_i + \frac{1}{n}\left[R_i(C) - K_i(C)\right]$$

Substituting (4.20) into (4.17) gives:

(4.21) $$d_{ij} = \frac{c_{ij} + c_{ji}}{2} + \frac{1}{2n}\left[(R_i(C) - K_i(C)) + (R_j(C) - K_j(C))\right] - (b_i + b_j) \quad \forall i, j; i \neq j$$

and finally substituting (4.20) and (4.21) into (4.15) gives

(4.22) $$e_{ij} = \frac{c_{ij} - c_{ji}}{2} - \frac{1}{2n}\left[(R_i(C) - K_i(C)) - (R_j(C) - K_j(C))\right] \quad \forall i, j; i \neq j$$

From the last equation it easily seen that $e_{ij} = -e_{ji}$

QED

Corollary 4.9

If $\forall i; R_i(C) = K_i(C)$ *then the decompositions in theorem 4.2 and 4.3 coincide*

Proof:

Follows directly from the results in theorem 4.3 and by choosing the *b*'s to be equal to zero.

QED

The following corollary is the promised part *(d)* of theorem 3.2, and enables us to decide whether an asymmetric matrix is Hamiltonian symmetrical or not, simply by comparing the differences between the symmetrical placed elements in the matrix, and the rows' and columns' sums.

Corollary 4.10

An asymmetric matrix C is Hamiltonian symmetrical iff

$$c_{ij} - c_{ji} = \frac{1}{n}\left[(R_i(C) - K_i(C)) - (R_j(C) - K_j(C))\right] \quad \forall i, j; i \neq j$$

Proof:

The result follows directly from *(iii)* in theorem 4.3 by claiming that the matrix $E = [0]$.

QED

Corollary 4.11

For any Hamiltonian cycle φ *we have that* $E(\vec{\varphi}) = \dfrac{C(\vec{\varphi}) - C(\overleftarrow{\varphi})}{2}$

Proof:

By the theorem 4.3 we have that

$$e_{ij} = \frac{c_{ij} - c_{ji}}{2} - \frac{1}{2n}\left[(R_i(C) - K_i(C)) - (R_j(C) - K_j(C))\right] \quad \forall i, j; i \neq j$$

The last part of the expression for the *e*-elements is constructed in the same way as the elements in a constant-TSP matrix. This means that adding all the elements corresponding to any Hamiltonian cycle, we will get all the row sums twice and all the column sums twice, but with different signs. Since the sum of all the row sums equals the sum of all the column sums in any matrix, the result follows.

QED

Theorem 4.2 and 4.3 give two different decompositions of an asymmetric matrix *C*. In order to illustrate the differences between the two decompositions a small example is offered in chapter 5.

Note

For skew symmetric matrices, the cost of any asymmetric pair will be zero. As a consequence, the optimal cost in any skew-symmetric matrix is non-positive. Further, the optimal cycles for the symmetric matrices in theorem 4.2 or 4.3 can give good upper bounds for TSP in the underlying asymmetric matrix, since we have that

$$A(\psi^*) \leq D(\varphi^*) + \min\left\{E(\varphi^*), E(\overleftarrow{\varphi^*})\right\} \leq D(\varphi^*)$$

Further, the decompositions made in theorem 4.2 and 4.3 can be used for finding lower bounds for the original asymmetric matrix, by using the lower bounds created by the saving matrices discussed and described in chapter 2. But first let E' be the symmetric matrix obtained by setting

$$e'_{ij} = \min\left\{e_{ij}, e_{ji}\right\}$$

This gives a symmetric matrix where all the entries are non-positive, since E is skew-symmetric. Any graph in E' will then have a lower cost than the same graph in E.

From the results in chapter 2 we then have:

$$C(\psi^*) = K(\psi^*) + D(\psi^*) + E(\psi^*) \geq K(\psi^*) + LB_D + LB_E \geq$$
$$K(\psi^*) + 2R_d - Mast(S_d) + (Mist + 1)(E^{*\prime})$$

where ψ^* is the optimal cycle for TSP in C. The right hand side of the inequality above can give good lower bounds, just by using simple lower bounds based on symmetric matrices only.

Chapter 5

In this chapter some examples are given to illustrate some of the results in the previous chapters. The examples are not very large and may not be representative for all possible situations encounter in connection with TSP, but are meant as tools for understanding and indications of what can be obtained. Most of the results of the previous chapters are illustrated, but not everything. Further, ten different instances of ATSP taken from a public library in order to test the new procedure described in chapter 4 are investigated. In the last section some final remarks are given as well as some suggestions for further research.

5.1 Examples Related to Chapter 2

In this section most of the results found in chapter 2 are illustrated with small examples. Many of the examples will be interlocked.

***Example 5.1.1** Traditional bounds*

We will start with a simple symmetric matrix given in table 5.1.1. The main objective will be to illustrate the different lower bounds discussed in chapter 2.

Table 5.1.1 Symmetric matrix *D*

Node	1	2	3	4	5	6	7	8
1	*	5	10	0	42	56	42	35
2	5	*	0	0	27	32	27	16
3	10	0	*	0	28	39	30	41
4	0	0	0	*	25	36	24	31
5	42	27	28	25	*	11	0	12
6	56	32	39	36	11	*	5	0
7	42	27	30	24	0	5	*	6
8	35	16	41	31	12	0	6	*
Sum	190	107	148	116	145	179	134	141

Starting with finding some traditional lower bounds for the problem, the results are summed up in Table 5.1.2

Table 5.1.2 Traditional lower bounds for matrix *D*

Type	Assignment*	MIST	MIST+1	LB0	LB1	LB2
Value	0 (22)	21	21	0	0	16

*) Smallest assignment found by model 1 is 0, with model 2, 22.

As can be seen from table 5.1.2, 21 is the best lower bound so far. The minimal spanning 1-tree is illustrated in figure 5.5.1. The dashed line is the smallest unused edge added to the minimal spanning tree. The two simple lower bounds LB0 and LB1 are zero, and the slightly more sophisticated bound LB2 is 16, see section 2.5.

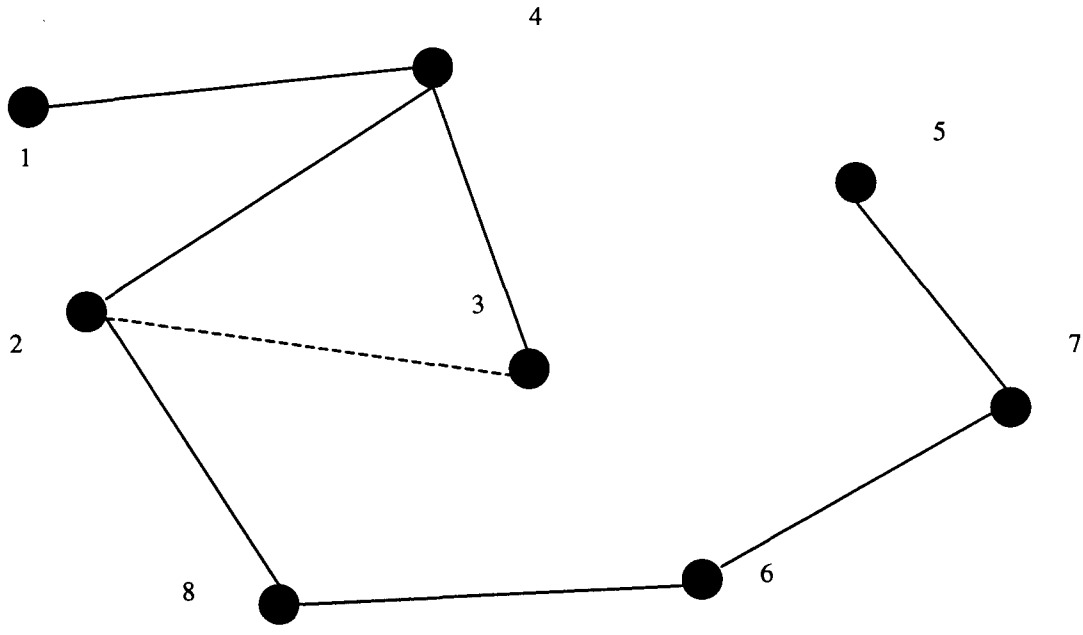


Figure 5.1.1

The lower bounds in table 5.1.2 can usually be improved by deleting one node in the graph, making the MIST among the remaining nodes and add the two cheapest edges from the deleted node, by this creating a 1-tree in the graph. The results are shown in table 5.1.3 below.

Table 5.1.3 1-trees created by deleting a node in the graph.

Deleted node	1	2	3	4	5	6	7	8
Value	26	34	39	26	32	27	32	35

As can be seen from table 5.1.3 the lower bounds have improved substantially compared with those in the previous table, the best being 39, found by first deleting node 3.

Example 5.1.2 *Bounds based on saving matrices*

However, making use of the saving matrices, we obtain other values as lower bounds. Since we can make similar bounds as in Table 5.1.2 for all eight saving matrices we get a lot of different bounds. The maximal assignment corresponding to each of the saving matrices will – after re-calculating the value to original cost matrix – as mentioned in chapter 2, be the same as in table 5.1.2. The column *2R –MAST(S)* in table 5.1.4 is based on the lower bound in lemma 2.1. An overview of all the saving matrices can be found in the appendix.

Table 5.1.4 Lower bounds for matrix D based on saving matrices.

Node	*LB3*	*LB4*	*LB5*	*2R – MAST(S)*
1	- 364	- 60	- 38	33
2	- 218	- 6	33	50
3	- 344	- 38	- 9	42
4	- 304	- 22	15	49
5	- 246	- 20	7	47
6	- 378	- 49	- 31	33
7	- 260	-20	2	46
8	- 294	- 30	-14	42

The lower bounds found by *LB3* are very bad, as are those for *LB4*. The situation is more mixed when we look at *LB5*. Here we find a good lower bound by node 2. This bound is better than any of the lower bounds found directly from the original cost matrix given in table 5.1.2 and competes reasonable well with the bounds found by the slightly more computationally complicated bounds in table 5.1.3. Real good lower bounds are found in every case in the last column. The best is 50. This lower bound is obtained when node 2 is chosen as the depot node. Except for node 1 and 6, the bounds in the last column are better than any bounds in table 5.1.3. Note also that the lower bounds for each choice of depot node becomes better and better.

The maximal spanning tree in saving matrix no. 2 is illustrated in figure 5.1.2 below.

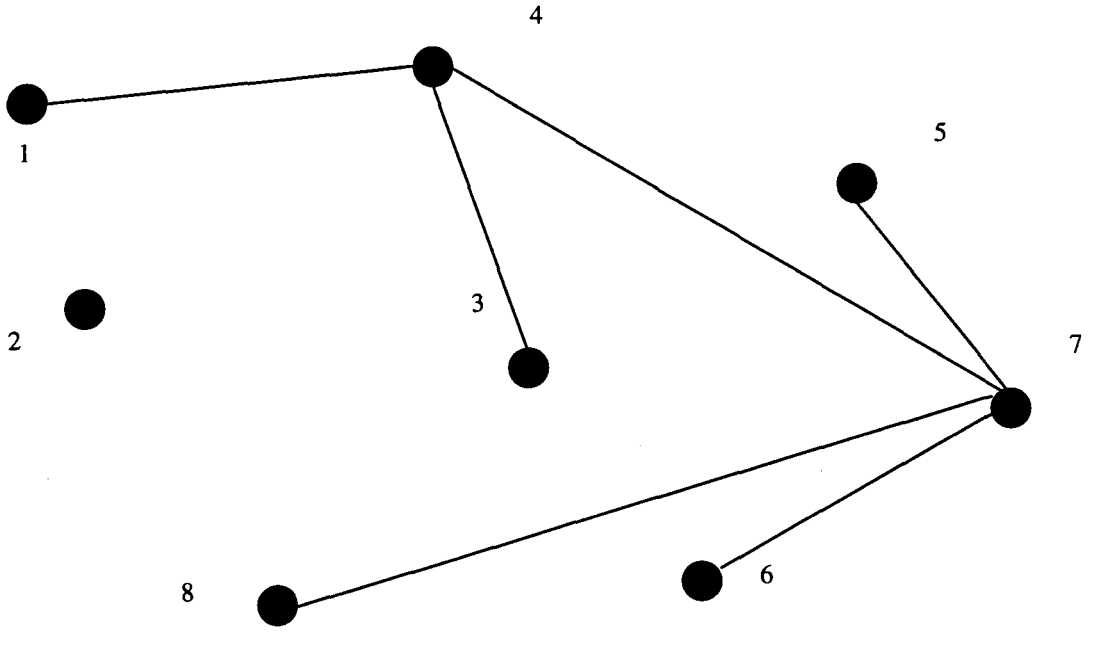


Figure 5.1.2

Example 5.1.3 *Bounds based on saving matrices and deletion of nodes*

We can get several more lower bounds for the matrix D by combining the technique used in table 5.1.3 with the lower bounds given by lemma 2.1. For each of the eight saving matrices, we delete one node at a time and calculate the resulting 1-tree. The results are shown in table 5.1.5.

Table 5.1.5 Bounds based on saving matrices and 1-trees.

Saving matrix	Deleted nodes							
	1	2	3	4	5	6	7	8
1	33	9	9	15	-51	-25	-51	-38
2	50	50	50	54	2	13	-7	13
3	32	35	42	32	-25	-22	-21	6
4	49	41	50	49	-4	-3	-4	7
5	-1	-4	-8	-6	47	41	41	47
6	-20	-39	-42	-42	11	33	16	17
7	-3	-7	-9	-8	46	46	46	44
8	-24	-7	-5	-24	25	41	25	42

Note that the values on the main diagonal are identical with lower bounds found in the last column in table 5.1.4, since these values are obtained by calculating the 1-trees in a saving matrix where the depot node is deleted.

From the table we see that the bounds vary very much. Good, new bounds are obtained for each saving matrix with exception of saving matrix no. 1 and 6. Better or identical bounds are obtained for saving matrix no. 2, 4, 5, and 7. The best bound is obtained by using saving matrix no 2 and deleting node 4. The corresponding 1-tree in the saving matrix is illustrated in figure 5.1.3 below. Note that the 1-tree in figure 5.1.4 is very close to become a Hamiltonian cycle. The only nodes with degrees different from 2, are node 8 with degree 1 and node 2 with degree 3.

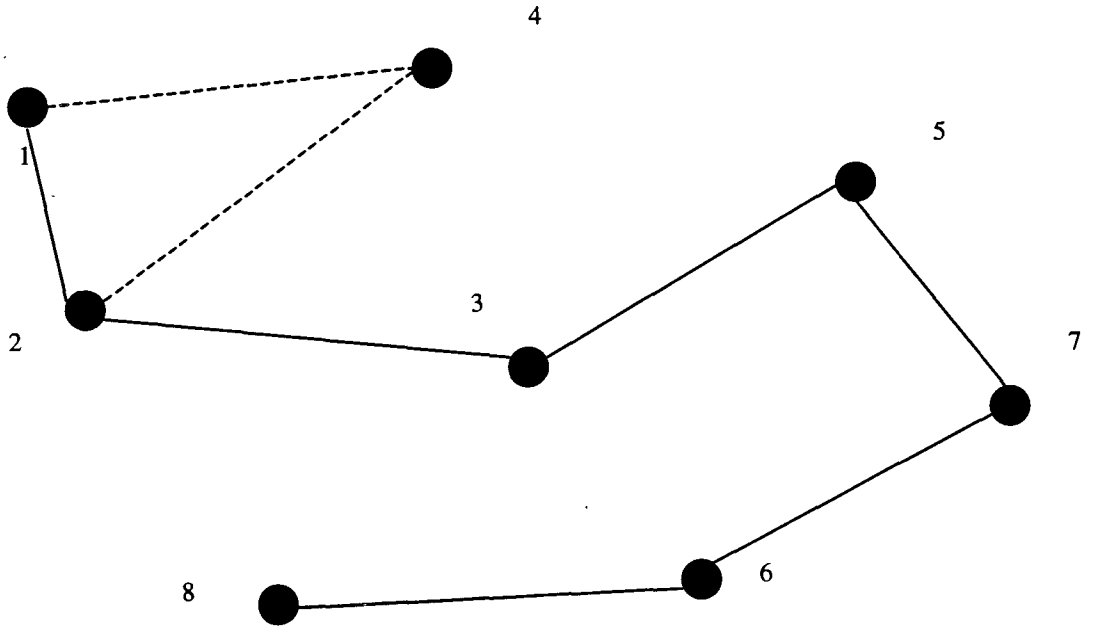


Figure 5.1.4

The example shows that there can be some virtue in taking the added computational burden of finding all the *nxn* 1-trees.

Example 5.1.4 *Bounds based on combinations of different upper bounds for the saving matrices.*

In this example model (2.30) – (2.33) will be applied systematically. As noted in connection with this model in chapter 2, there is no need to use the model *nxn* times. All lower bounds obtainable by this model can be found by choosing any of the saving matrices in the objective and keeping this matrix in the objective. Then, choosing the *n* different upper bounds corresponding to the lower bounds in the last column in table 5.1.4 will give all obtainable lower bounds with this method.

In table 5.1.6 below this is done. In addition, the calculations are repeated for all choices of saving matrices in the objective. The reason for this will be explained after table 5.1.6. In table 5.1.6 the second column corresponds to the lower bounds for the TSP in question found in table 5.1.4 and the third column gives the upper bounds in the corresponding saving matrices. The rest of the entries in the table are lower bounds for the TSP given a specific upper (or lower) bound for a specific choice of the objective. The upper bounds corresponding to the costs measured in the saving matrices, can all be very different. These upper bounds are not shown in table 5.1.6.

Table 5.1.6 Lower bounds for TSP in matrix D based on model (2.30) –(2.33)

	Lower bound in D	*MAST* in S^i	Saving matrix chosen in the objective in the model							
			1	2	3	4	5	6	7	8
1	33	347	34	34	34	34	34	34	34	34
2	50	164	54	54	54	54	54	54	54	54
3	42	254	44	44	44	44	44	44	44	44
4	49	183	49	49	49	49	49	49	49	49
5	47	243	49	49	49	49	49	49	49	49
6	33	325	34	34	34	34	34	34	34	34
7	46	222	49	49	49	49	49	49	49	49
8	42	240	44	44	44	44	44	44	44	44

Table 5.1.6 shows that all the previous known lower bounds are improved by this technique. It also shows that the best lower bound is 54. This value is obtained by using any of the saving matrices as input in the objective and using saving matrix no. 2 in restriction (2.33). The rest of the table shows what the theory told us would happen, namely that the obtained lower bounds are equal for each choice of the input matrix in restriction (2.33) whatever saving matrix is used in the objective. Seemingly, a lot of unnecessary work has been done in table 5.6. However, the 8 different cases, all giving 54 as the value for the lower bound, are all assignments, but can be different assignments with equal values. In the table these assignments are as follows:

For saving matrices 1,7, and 8 we have the assignment

$$1-3-1\ ;\ 2-4-2\ ;\ \text{and}\ 5-6-7-8-5$$

For saving matrix no. 6 we have the assignment

$$1-3-1\ ;\ 2-4-2\ ;\ 5-6-5\ ;\ \text{and}\ 7-8-7$$

For the saving matrices 2, 3, 4, and 5 we have the assignment

$$1-2-8-6-7-5-3-4-1$$

which is a Hamiltonian cycle.

Hence, we have solved the TSP only by using simple upper and lower bounds and solving assignment problems with a knap-sack restriction added.

If one tries to find the optimal solution directly by for example applying model 2 to the matrix in table 5.1.1, the same optimal solution is found after adding two sub-tour eliminating restrictions in one batch. It is not necessary to use IP.

In the next examples, we will use the TSP instance in table 5.1.1 as input to different heuristics. This will be done by applying the original matrix directly to the heuristics and then using the different saving matrices as well.

Example 5.1.5 *Nearest neighbour (NN)*

In this example the naïve heuristic nearest neighbour is applied to the original cost matrix *D* and each of the saving matrices for the matrix. The costs of the resulting Hamiltonian cycles are displayed in table 5.1.7 below. Since some of the matrices contain identical values on some rows the following rule of performance has been used when the heuristic is applied: If one has a choice between two or more nodes, the node with the lowest index is chosen.

For the saving matrices, the heuristic is not applied to the starting node corresponding to the depot node since any saving matrix has only zeros in this row.

Table 5.1.7 Costs of Hamiltonian cycles for the nearest neighbour heuristic

Applied matrix	Starting node							
	1	2	3	4	5	6	7	8
D	68	63	88	64	64	75	77	76
SAV1	-	95	82	79	80	89	87	68
SAV2	61	-	96	69	54	77	62	76
SAV3	75	56	-	54	71	75	77	76
SAV4	89	56	56	-	74	75	77	78
SAV5	83	77	76	74	-	54	54	71
SAV6	67	54	83	73	63	-	86	67
SAV7	63	56	59	69	59	54	-	67
SAV8	76	63	78	78	54	77	54	-

As can bee seen from the table above, applying *NN* to the original cost matrix give us a cycle with cost 63 as the best result. This cycle is: 2 – 3 – 4 – 1 – 5 – 7 –6 – 8 – 2. Applying *SAV1* does not yield any better results, but a sequence of rather bad cycles. *SAV4* on the other hand gives two instances with cost 56. (It is the same cycle generated for this matrix starting either with node 2 or 3). For all the other saving matrices we get cycles with cost 54. From table 5.1.5 we know that 54 is a lower bound for the TSP in question and we can conclude that *NN* finds the optimal solution when applied to 6 of the eight saving matrices, but not to the original cost matrix. In addition several good, but non-optimal solutions are found by the saving matrices, which are better than those found with the original cost matrix.

Example 5.1.6 *The Clarke and Wright heuristic*

Applying the classical Clarke and Wright heuristic to the 8 saving matrices gives the following result:

Table 5.1.8 Clarke and Wright heuristic applied on *D*

Matrix	*Sav1*	*Sav2*	*Sav3*	*Sav4*	*Sav5*	*Sav6*	*Sav7*	*Sav8*
Cost	54	56	71	56	69	56	69	62

Table 5.1.8 shows that this classical heuristic finds the optimal solution in one out of eight cases and finds very good solution in three other cases.

In chapter 2 it was mentioned that since the saving matrices are as good inputs to the TSP as the original cost matrix, we may as well use the Clarke and Wright algorithm directly on the original cost matrix. Doing this we get the cycle: 1 – 4 – 2 – 3 – 5 – 6 – 8 – 7 – 1, with cost 76.

Example 5.1.7 *Nearest insertion*

Again we first apply the original matrix 8 times to this heuristic, and then we do the same for each of the saving matrices. It is again a bit awkward to choose the depot node as a starting node for a given saving matrix. Hence, this possibility is left out in table 5.1.9 below, showing the results of all the other applications.

Table 5.1.9 Nearest insertion heuristic

Applied matrix	Starting node							
	1	2	3	4	5	6	7	8
D	67	67	67	67	56	61	62	56
SAV1	-	61	73	70	61	75	75	75
SAV2	74	-	74	74	54	54	54	54
SAV3	54	54	-	54	54	61	61	61
SAV4	67	54	56	-	61	61	61	61
SAV5	67	67	56	67	-	67	56	67
SAV6	69	69	69	69	69	-	55	63
SAV7	67	67	67	67	69	56	-	67
SAV8	74	74	74	74	74	56	54	-

As can be seen from table 5.1.9, the nearest insertion heuristic does not find an optimal solution when the original cost matrix is applied. The best cycle is found starting with node 6 and giving the cycle: 6 – 5 – 7 – 4 – 1 – 3 – 2 – 8 – 6 giving a cost of 61.Neither is an optimal solution found when the saving matrices based on node 1, 5, 6, and 7 are used, but for the three latter ones, better solutions are found than for the original matrix. For the other saving matrices an optimal solution is found.

Example 5.1.8 *Savings and product matrices*

This example refers to sub-section 2.7.1. In table 5.1.10 an asymmetric matrix is given.

Table 5.1.10 An asymmetric matrix

Node	1	2	3	4	5	6	7	8
1	-	40	35	70	20	45	80	25
2	30	-	9	16	-13	12	75	48
3	15	-1	-	-11	-37	-12	55	32
4	55	11	-6	-	-33	-8	75	68
5	60	37	23	22	-	24	95	76
6	25	2	-12	-13	-36	-	56	41
7	70	75	65	80	45	70	-	90
8	50	83	77	108	61	86	125	-

Making the saving matrix based node 1 as the depot from the matrix in table 5.1.10, we get the matrix in table 5.1.11

Table 5.1.11 A saving matrix based on the matrix in table 5.1.10

Node	1	2	3	4	5	6	7	8
1	-	0	0	0	0	0	0	0
2	0	-	56	84	63	63	35	7
3	0	56	-	96	72	72	40	8
4	0	84	96	-	108	108	60	12
5	0	63	72	108	-	81	45	9
6	0	63	72	108	81	-	45	9
7	0	35	40	60	45	45	-	5
8	0	7	8	12	9	9	5	-

Since the saving matrix in table 5.1.11 is symmetric, we know that the original matrix is Hamiltonian symmetric. Moreover, the saving matrix can be shown to be a symmetric product matrix defined by the vector $\boldsymbol{d}$ = [0, 7, 8, 12, 9, 9, 5, 1]. This vector can be found by first observing that since the first row consists only of zeros, the first element in such a vector must be zero as well. The rest of the elements in the vector can be found by solving a set of equations for example using the six last entries in row 2. This gives the following six equations:

$$d_2 d_j = s_{2j}; j = 3,4,5,6,7,8$$

Hence, the maximal Hamiltonian cycle can be found in polynomial time and as a consequence, the minimal cycle in the original matrix.

Example 5.1.9 *Savings and circulant matrices*

This example refers to sub-section 2.7.2. In table 5.1.12 is given a set of values for the first row and column and the chosen values for the *c's* in order to calculate the rest of the elements of the matrix *C*.

Table 5.1.12 Starting values for the matrix *C*

j,k	1	2	3	4	5	6	7	8	9	10	11
1. row	0	9	16	13	20	11	30	17	15	24	18
1. col.	0	12	24	18	17	35	21	14	19	27	33
c	-	17	5	9	11	7	18	6	20	15	-

Using the values given in Table 5.1.12 we get the full *11x11* matrix *C* in table 5.1.13 and the corresponding saving matrix based on node 1 in Table 5.1.14.

Table 5.1.13 The resulting matrix C

Node	1	2	3	4	5	6	7	8	9	10	11
1	0	9	16	13	20	11	30	17	15	24	18
2	12	-	11	20	23	11	35	11	21	16	15
3	24	18	-	20	39	26	43	34	21	42	22
4	18	7	19	-	21	24	39	24	26	24	30
5	17	20	13	15	-	11	42	25	21	34	17
6	35	26	45	28	40	-	48	47	41	48	46
7	21	23	19	28	21	17	-	21	31	36	28
8	14	12	23	9	28	5	29	-	12	33	23
9	19	19	24	25	17	24	29	21	-	26	32
10	27	31	34	29	40	20	51	24	27	-	28
11	33	25	44	42	42	37	45	44	28	42	-

Table 5.1.14 The saving matrix for Table 5.1.13, based on node 1 as a depot.

Node	1	2	3	4	5	6	7	8	9	10	11
1	-	0	0	0	0	0	0	0	0	0	0
2	0	-	17	5	9	11	7	18	6	20	15
3	0	15	-	17	5	9	11	7	18	6	20
4	0	20	15	-	17	5	9	11	7	18	6
5	0	6	20	15	-	17	5	9	11	7	18
6	0	18	6	20	15	-	17	5	9	11	7
7	0	7	18	6	20	15	-	17	5	9	11
8	0	11	7	18	6	20	15	-	17	5	9
9	0	9	11	7	18	6	20	15	-	17	5
10	0	5	9	11	7	18	6	20	15	-	17
11	0	17	5	9	11	7	18	6	20	15	-

As can be seen, the matrix in Table 5.1.14 is almost circulant. Now, applying Theorem 2 on the circulant part of the saving matrix, starting with node 2 and using the rule "farthest from" yields the Hamilton path:

2 - 10 - 8 - 6 - 4 - 5 - 3 - 11 - 9 - 7

with total cost in the saving matrix equal to 177. The first row sum of Table 2 equals 173 and the first column sum equals 220. By equation (2.3) the optimal TSP for C has the cost 216 and is:

1 - 2 - 10 - 8 - 6 - 4 - 5 - 3 - 11 - 9 - 7 - 1

One can obtain different optimal Hamiltonian cycles by starting the process by choosing some of the other nodes as a starting node.

Example 5.1.10 *The new class of polynomial solvable TSP*

This example refers to sub-section 2.7.5. Let the vector $\boldsymbol{a}$ = [0, 6, 4, 10, 2, 3, 5, 9] and $\boldsymbol{b}$ = [0, 3, 1, 7, 12, 9, 8, 6]. This gives the following matrix where the entries below the main diagonal are formed as in a constant-TSP matrix with the help of the two

vectors. The entries above the main diagonal is chosen randomly and can in principle take any values.

Table 5.1.15 A partly constant-TSP matrix

Node	1	2	3	4	5	6	7	8
1	0	6	4	10	2	3	5	9
2	3	0	15	13	25	12	6	14
3	1	7	0	20	13	18	25	4
4	7	13	11	0	16	27	19	12
5	12	18	16	22	0	22	17	10
6	9	15	13	19	11	0	26	16
7	8	14	12	18	10	11	0	15
8	6	12	10	16	8	9	11	0

Making the saving matrix based on node 1 as the depot node gives the following matrix:

Table 5.1.16 Saving matrix for the partly constant-TSP matrix

Node	1	2	3	4	5	6	7	8
1	0	0	0	0	0	0	0	0
2	0	0	-8	0	-20	-6	3	-2
3	0	0	0	-9	-10	-14	-19	6
4	0	0	0	0	-7	-17	-7	4
5	0	0	0	0	0	-7	0	11
6	0	0	0	0	0	0	-12	1
7	0	0	0	0	0	0	0	2
8	0	0	0	0	0	0	0	0

The matrix in table 5.1.16 clearly is an upper triangular matrix and the maximal TSP in the saving matrix can be found in polynomial time. Hence, we have found the minimal TSP in the original matrix.

Example 5.1.11 *TSP with identical costs for different matrices for each Hamiltonian cycles.*

In section 2.9 it was proved that there exists an infinite number of matrices which have the same cost for every cycle. Now let A be the symmetric product matrix constructed by the vector [10, 7, 8, 12, 9, 9, 5,1] giving the matrix in table 5.1.17.

Table 5.1.17 The symmetric product matrix *A*

Node	1	2	3	4	5	6	7	8	Sum
1	-	70	80	120	90	90	50	10	510
2		-	56	84	63	63	35	7	378
3			-	96	72	72	40	8	424
4				-	108	108	60	12	588
5					-	81	45	9	468
6						-	45	9	468
7							-	5	280
8								-	60

Remember that the matrix *A* can be solved in polynomial time.

From table 5.1.17 we get the saving matrix based on node 1 as the depot in the next table, 5.1.18.

Table 5.1.18 The saving matrix *S(A)* based on node 1 as the depot

Node	1	2	3	4	5	6	7	8
1	-	0	0	0	0	0	0	0
2		-	94	106	97	97	85	73
3			-	104	98	98	90	82
4				-	102	102	110	118
5					-	99	95	91
6						-	95	91
7							-	55
8								-

Now, let *B* be the symmetric matrix in table 5.1.19. Note that some of the entries in the matrix are negative, whilst all the row sums are positive numbers. Hence, the matrix *B* is not a symmetric product matrix.

Table 5.1.19 The symmetric matrix *B*

Node	1	2	3	4	5	6	7	8	Sum
1	-	30	50	80	30	90	60	140	480
2		-	-4	14	-27	33	15	107	168
3			-	36	-8	52	30	118	274
4				-	18	78	40	112	378
5					-	31	5	89	138
6						-	65	149	498
7							-	155	370
8								-	870

Based on the matrix *B*, we calculate the saving values for the same depot as we did for *A*. The result is given in table 5.1.20.

Table 5.1.20 The saving matrix *S(B)* based on node 1 as a depot node

Node	1	2	3	4	5	6	7	8
1	-	0	0	0	0	0	0	0
2		-	84	96	87	75	63	94
3			-	94	88	88	80	72
4				-	92	92	100	108
5					-	89	85	81
6						-	85	81
7							-	45
8								-

Now, using lemma 2.2 and comparing the entries in the two saving matrices *S(A)* and *S(B)*, one observes that the difference between the entries are always equal to 10, except for the first row, ie $k = 10$. Further, the difference between the first row sum in the two matrices *A* and *B*, is 30. Since we have symmetric matrices in our cases, we shall take two times this difference, ie 60. On the other hand, with 8 nodes we have that $k(n-2) = 60$. By lemma 2.2 we then have that the two matrices have the same costs for the same cycles.

5.2 Examples Related to Chapter 3

This section deals with examples related to the spectral theorem.

***Example 5.2.1** Decomposition of a symmetric matrix*

We will use the symmetric matrix *D* from table 5.1.1 to illustrate how such a matrix can be decomposed into 8 polynomial product matrices with the help of the spectral theorem. Further, corollary 3.5 is applied in order to get upper and lower bounds for the TSP in the matrix *D*.

The eigenvalues of the matrix are given in table 5.2.1 below.

Table 5.2.1 Eigenvalues for the matrix in table 5.1.1

Eigen-value no.	1	2	3	4	5	6	7	8
Value	-0.1846	3.4072	-3.8778	-12.3608	12.6973	-25.0599	-124.3386	149.7172

The dominating eigenvalue is eigenvalue no. 8 and it is a positive number as it should be according to the theorem of Perron – Frobenius. The sum of the eigenvalues is zero. The corresponding eigenvectors are given in table 5.2.2

Table 5.2.2. Eigenvectors for the matrix in table 5.1.1

Co-ord.no.	Eigenvector number, corresponding to the eigenvalues in table 5.2.1							
	1	2	3	4	5	6	7	8
1	0.2220	0.2130	0.3007	0.5223	0.1998	0.2903	-0.4576	0.4566
2	-0.1340	-0.0510	-0.6490	-0.3351	0.5002	0.2331	-0.2701	0.2619
3	0.1764	0.3839	0.0307	-0.5013	-0.4077	-0.3868	-0.3506	0.3612
4	-0.4738	-0.6979	0.1363	0.0516	-0.2394	-0.1531	-0.3284	0.2802
5	0.4806	-0.2777	-0.3662	0.3257	0.0482	-0.4954	0.2885	0.3475
6	0.1251	-0.1970	0.5010	-0.4479	0.3088	0.1521	0.4419	0.4245
7	-0.6558	0.4480	-0.0188	0.2243	0.1435	-0.2963	0.3281	0.3201
8	-0.0222	0.0292	-0.2889	0.0632	-0.6067	0.5761	0.3174	0.3324

Note that vector no. 8 is a positive vector as predicted by the Perron – Frobenius theorem.

Each of the vectors can now be used to construct a symmetric polynomial solvable product matrix. If the corresponding eigenvalue is positive, one shall minimise. If the eigenvalue is negative, one shall maximise. According to the alternative proof for theorem 1.10 in chapter 1, the cycle giving the minimal cost is

$$1-8-2-6-4-5-3-7-1$$

and the cycle giving the maximal cost is

$$1-2-4-6-8-7-5-3-1$$

if the entries in the underlying vector form a descending sequence. Since the entries of the eigenvectors in table 5.2.2 do not form such descending sequences, we have to rearrange the entries. The results are shown in table 5.2.3.

Table 5.2.3 Optimal cycles and bounds for each of the product matrices

Matrix	Opt	Cycle φ_k^*	$D(\varphi_k^*)$	$T_k(\varphi_k^*)$	$\lambda_k T_k(\varphi_k^*)$
1	Min	5 – 1 – 6 – 2 – 7 – 4 – 8 – 3 – 5	281	0.6077	-0.1122
2	Min	7 – 4 – 3 – 6 – 8 – 2 – 1 – 5 – 7	126	-0.8331	-2.8888
3	Max	6 – 1 – 3 – 8 – 2 – 5 – 7 – 4 – 6	210	0.6488	-2.5158
4	Max	1 – 5 – 8 – 2 – 3 – 6 – 4 – 7 – 1	211	0.6677	-8.2527
5	Min	2 – 8 – 6 – 4 – 7 – 5 – 1 – 3 – 2	128	-0.7899	-10.0296
6	Max	8 – 1 – 6 – 7 – 5 – 3 – 4 – 2 – 8	140	0.6620	-16.5892
7	Max	6 – 7 – 5 – 4 – 1 – 3 – 2 – 8 – 6	56	0.5948	-73.9613
8	Min	1 – 2 – 6 – 7 – 5 – 8 – 3 – 4 – 1	95	0.9426	+141.1229
Sum	-	-	-	-	+26.7733

As can be seen from table 5.2.3, the upper bounds are in most cases very poor. Only one good bound is found. We get a lower bound of 27, which is better than the assignment bounds, and the bounds found by MIST and MIST+1. This new bound can also compete with some of the bounds found in table 5.1.3, namely 1-trees created

by deleting a node from the graph, but the best of the latter ones are much better than 27.

Example 5.2.2 *New class of polynomial solvable STSP*

Now let A be the symmetric product matrix defined in example 5.1.10 and given in table 5.1.15. Further let B be the symmetric product matrix given by the vector $\boldsymbol{b}$ = [17, 5, 8, 20, 15, 10, 4, 3]. The matrix $C = A + B$, then becomes the matrix in table 5.2.4.

Table 5.2.4 A matrix composed of the sum of two symmetric product matrces

Node	1	2	3	4	5	6	7	8	Sum
1	389	155	216	460	345	260	118	61	2004
2		74	96	184	138	113	55	22	837
3			128	256	192	152	72	32	1144
4				544	408	308	140	72	2372
5					306	231	105	541	1779
6						181	85	391	1369
7							41	17	633
8								10	307

It is easily checked that this matrix is not a symmetric product matrix. Hence, it cannot be solved by the polynomial algorithm described in chapter 1 concerning symmetric product matrices. However, ordering the co-ordinates of vector $\boldsymbol{a}$ in decreasing order we will get 4, 1, 5, 6, 3, 2, 7, 8. Doing the same with the co-ordinates for vector $\boldsymbol{b}$, we get the same order. This means that the two components of the matrix obtain their optima by the same cycle. This cycle can be found by the algorithm in chapter 1.

Example 5.2.3 *Asymmetric matrices and the spectral theorem*

We will start with the asymmetric matrix in table 5.2.5.

Table 5.2.5 An asymmetric matrix A

Node	1	2	3	4	5	6	7	8
1	-	22	16	18	48	62	60	85
2	19	-	12	24	41	44	48	72
3	18	10	-	15	30	45	43	89
4	16	26	14	-	42	50	47	91
5	48	38	34	40	-	16	12	59
6	62	49	43	55	15	-	18	48
7	56	48	42	48	12	17	-	62
8	87	72	89	91	60	48	63	-

The eigenvectors for this matrix can be found in table 5.2.6

Table 5.2.6 Eigenvectors for the matrix in table 5.2.5

	Eigenvalue number							
	1	2	3	4	5	6	7	8
Real part	320.06	-158.59	-85.74	-27.02	-10.67	-10.67	-18.81	-10.55
Imag. part	0	0	0	0	1.91	-1.91	0	0

Note that six of the eigenvalues are real numbers. The last two are complex conjugate numbers of each other. The sum of the eigenvalues is - of course – still zero.

The eigenvectors are given in table 5.2.7

Table 5.2.7 Eigenvectors for the matrix in table 5.2.5

Co-ordinate number		Eigenvector number							
		1	2	3	4	5	6	7	8
1	Real part	0.3611	-0.4007	0.2549	-0.2767	-0.1931	-0.1931	0.5132	-0.4782
	Imag. part	0	0	0	0	-0.4266	0.4266	0	0
2	Real part	0.3061	-0.2905	0.1362	0.6507	0.1104	0.1104	0.0517	-0.1643
	Imag. part	0	0	0	0	-0.1777	0.1777	0	0
3	Real part	0.3068	-0.3768	-0.0845	0.3449	-0.0576	-0.0576	-0.0669	0.6363
	Imag. part	0	0	0	0	0.6061	-0.6061	0	0
4	Real part	0.3406	-0.3843	0.0019	-0.5408	0.1610	0.1610	-0.5961	0.0462
	Imag. part	0	0	0	0	0.0240	-0.0240	0	0
5	Real part	0.2896	0.1083	-0.4353	-0.0138	0.0176	0.0176	0.1493	0.1121
	Imag. part	0	0	0	0	0.1958	-0.1958	0	0
6	Real part	0.3262	0.2730	-0.3692	0.2147	-0.1326	-0.1326	-0.4478	-0.5192
	Imag. part	0	0	0	0	-0.4921	0.4921	0	0
7	Real part	0.3288	0.1776	-0.4913	-0.2023	0.0642	0.0642	0.3890	0.1396
	Imag. part	0	0	0	0	0.0594	-0.0594	0	0
8	Real part	0.5171	0.5895	0.5850	-0.0369	0.0235	0.0235	-0.0083	0.1892
	Imag. part	0	0	0	0	0.1824	-0.1824	0	0

As can be seen from table 5.2.7 all the eigenvectors are real except no. 5 and 6. These two eigenvectors are complex conjugate of each other.

By corollary 3.10 there exists 32 real vectors $\vec{b_k}, \vec{d_k}, \vec{\mu_k}$ and $\vec{\rho_k}$ *such that*

$$A = \sum_k \mu_k B_k + \sum_k \mu_k D_k + \sum_k \rho_k E_k$$

where B_k and D_k are polynomial solvable symmetric product matrices $[b_{k_s} b_{k_t}]$ and $[d_{k_s} d_{k_t}]$, respectively, and E_k are skew-symmetric matrices $[b_{k_s} d_{k_t} - b_{k_t} d_{k_s}]$.

The vectors $\vec{b_k}, \vec{d_k}, \vec{\mu_k}$ and $\vec{\rho_k}$ are the real part of the eigenvector k, the imaginary part of the eigenvector k, the real part of the eigenvalue k, and the imaginary part of the eigenvalue k, respectively. From the two previous tables, we have that $\vec{b_5} = \vec{b_6}$ and $\vec{d_5} = -\vec{d_6}$. Moreover, $\vec{\rho_k}$ and $\vec{d_k}$ are zero vectors for all k except for 5 and 6. As a consequence, the matrix A can be decomposed in a much simpler way than

indicated by corollary 3.10, namely into 10 different matrices, where 9 is polynomial solvable product matrices and the last one is a skew- symmetric matrix. The decomposition becomes:

$$A = \sum_{k=1}^{8} \mu_k B_k - 21.34[d_{5s}d_{5t}] + 3.9[b_{5s}d_{5t} - b_{5t}d_{5s}]$$

Example 5.2.4 *Hermitian matrices and the spectral theorem*

Again we will start with the asymmetric matrix in table 5.2.5. However, this time we decompose the matrix into the two following matrices, one symmetric given in table 5.2.8, and one skew-symmetric given in table 5.2.9.

Table 5.2.8 Symmetric component of the matrix in table 5.2.5

Node	1	2	3	4	5	6	7	8	Row sum
1	-	20.5	17	17	48	62	58	86	308.5
2		-	11	25	39.5	46.5	48	72	262
3			-	14.5	32	44	42.5	89	250
4				-	41	52.5	47.5	91	288.5
5					-	15.5	12	59.5	247.5
6						-	17.5	48	285.5
7							-	62.5	288
8								-	508

Table 5.2.9 The skew-symmetric component of the matrix 5.2.5

Node	1	2	3	4	5	6	7	8
1	-	1.5	-1	1	0	0	2	-1.5
2	-1.5	-	1	-1	1.5	-2.5	0	0
3	1	-1	-	0.5	-2	1.0	0.5	0
4	-1	1	-0.5	-	1	-2.5	-0.5	0
5	0	-1.5	2	-1	-	0.5	0	-0.5
6	0	2.5	-1	2.5	-0.5	-	0.5	0
7	-2	0	-0.5	0.5	0	-0.5	-	-0.5
8	1.5	0	0	0	0.5	0	0.5	-

Now, by using the last two matrices, one can construct a Hermitian matrix. The eigenvalues and the eigenvectors for this matrix are given in table 5.2.10 and 5.2.11, respectively

Table 5.2.10 Eigenvectors for the Hermitian matrix

	Eigenvalue number							
	1	2	3	4	5	6	7	8
Eigenvalue	320.09	-158.62	-85.84	-27.24	-4.12	-11.12	-15.47	-17.67

Note that all the eigenvalues are real numbers, as they should be for Hermitian matrices.

Table 5.2.7 Eigenvectors for the Hermitian matrix in table 5.2.11

Co-ordinate number		Eigenvector number							
		1	2	3	4	5	6	7	8
1	Real part	0.3590	-0.4028	0.2363	-0.1799	0.1054	-0.1589	-0.0583	0.0751
	Imag. part	0	-0.0295	-0.0363	-0.2284	-0.1164	-0.3238	-0.5408	-0.3317
2	Real part	0.3088	-0.2873	0.1319	0.4486	-0.4233	-0.1152	0.2694	-0.1580
	Imag. part	-0.0039	-0.0270	-0.0320	0.5289	-0.1677	-0.0284	0.0361	0.0408
3	Real part	0.3060	-0.3750	-0.0778	0.1280	0.5691	0.1195	-0.1897	0.1207
	Imag. part	-0.0005	0.0327	0.0085	0.2733	0.3826	0.3307	0.1469	-0.0369
4	Real part	0.3430	-0.3843	0.0101	0.2913	-0.2462	0.1167	0.0025	-0.0037
	Imag. part	-0.0037	-0.0322	-0.0161	-0.4331	-0.1091	0.0888	0.4216	0.4347
5	Real part	0.2904	0.1085	-0.4308	0.0084	-0.3010	0.5115	-0.2395	-0.1631
	Imag. part	-0.0020	.0081	0.1107	0.0137	0.2128	-0.3191	0.0585	-0.3451
6	Real part	0.3223	0.2638	-0.3527	0.1045	0.1040	-0.1751	-0.2639	0.3782
	Imag. part	0.0026	0.0295	0.0739	0.1428	-0.2313	-0-2249	-0.2140	0.5103
7	Real part	0.3321	0.1811	-0.4827	-0.1114	0.0924	-0.3649	0.4537	-0.2279
	Imag. part	-0.0047	0.0107	0-1190	-0.1600	-0.0650	0.3417	-0.0386	-0.2228
8	Real part	0.5157	0.5876	0.5706	-0.0248	0.0508	0.0913	0.0023	-0.0272
	Imag. part	-0.0003	0.0518	-0.1321	-0.0256	0.0989	0.0877	0.1065	-0.0362

By combining the different vectors constituting the eigenvalues and the real and imaginary parts of the eigenvectors, the original matrix can be decomposed into sixteen symmetric product matrices and eight skew-symmetric matrices, see theorem 3.12.

5.3 Examples Related to Chapter 4

Example 5.3.1 *A Hamiltonian symmetric matrix*

Let H be the asymmetric matrix in table 5.3.1. In addition to the cost elements, the row and column sums are calculated.

Table 5.3.1 The matrix H with row and column sums

Node	1	2	3	4	5	6	7	8	Sum
1	-	19	16	18	48	62	57	85	305
2	19	-	12	24	39	44	48	72	258
3	14	10	-	14	30	41	41	87	237
4	16	22	14	-	39	50	47	89	277
5	47	38	31	40	-	14	12	59	241
6	62	44	43	52	15	-	18	48	282
7	56	47	42	48	12	17	-	62	284
8	85	72	89	91	60	48	63	-	508
Sum	299	252	247	287	243	276	286	502	2392

Solving TSP for the matrix H with model 1, gives an optimal solution of 230 and one has to add 6 sub-tour eliminating restrictions. The problem can be solved as LP. It is difficult to realise that H is Hamiltonian symmetric just by looking at the elements in the matrix. However, solving a simple LP, one easily finds that H can be decomposed into to a sum of two other matrices, namely the symmetric matrix D in example 5.1.1, given in table 5.1.1 and the constant-TSP matrix K constructed from the two vectors $\vec{a}$ = [6,12,2,14,3,4,12,48] and $\vec{b}$ = [2,8,0,12,0,0,9,44]. The sum of the elements in these two vectors adds up to 176. Since an optimal solution in D can be found by adding only two sub-tour eliminating constraints and that an optimal solution to H and D coincides, this asymmetric TSP can be solved more easy as a symmetric TSP than as an asymmetric one.

In theorem 4.2 two other criteria are stated as tools for identifying Hamiltonian symmetric matrices. Calculating the saving matrix based on node 1 as a depot gives the following result.

Table 5.3.2 Saving matrix for H based on node 1 as a depot.

Node	1	2	3	4	5	6	7	8
1	-	0	0	0	0	0	0	0
2	0	-	23	13	28	37	28	32
3	0	23	-	18	32	35	30	12
4	0	13	18	-	25	28	26	12
5	0	28	32	25	-	95	92	73
6	0	37	35	28	95	-	101	99
7	0	28	30	26	92	101	-	79
8	0	32	12	12	73	99	79	-

As can be seen from table 5.3.2, the saving matrix is symmetric and hence H is Hamiltonian symmetric.

The last criterion in theorem 4.2 concerns the difference between two entries in the cost matrix placed symmetrically. In table 5.3.3 these differences are calculated for all the pairs of H.

Table 5.3.3 $h_{ij} - h_{ji}$ for all symmetric pairs in H.

Node	1	2	3	4	5	6	7	8
1	-	0	2	2	1	0	1	0
2		-	2	2	1	0	1	0
3			-	0	-1	-2	-1	-2
4				-	-1	-2	-1	-2
5					-	-1	0	-1
6						-	1	0
7							-	-1
8								-

For example the difference between the pair (3,6) and (6,3) can be seen to be –2. Calculating the right hand side of *(d)* in theorem 4.2 for $i = 3$ and $j = 6$ with the row and column sums given in table 5.3.1. we get

$$\frac{1}{8}((R_3 - K_3) - (R_6 - K_6)) = \frac{1}{8}((237 - 247) - (282 - 276)) = -2$$

The rest of the entries in table 5.3.3 can be checked in a similar way.

Example 5.3.2 *Optimal pairs of Hamiltonian cycles*

An optimal pair of Hamiltonian cycles is defined in chapter 4, definition 4.3. Theorem 4.4 says that finding such an optimal pair can be found by solving an appropriate symmetric TSP. Now, let the asymmetric matrix be A identical with the matrix given in table 5.2.5.

This matrix A is not Hamiltonian symmetric. Finding the optimal solution in the matrix A with model 1, requires 6 sub-tour eliminating restrictions and the cycle is

$$1 - 4 - 3 - 5 - 7 - 6 - 8 - 2 - 1$$

with cost 230. The reversed cycle has cost 237. Hence, the cost of this asymmetric pair has the cost 467.

Does a better asymmetric pair exist? In order to find the optimal pair, we construct the matrix be as indicated by theorem 4.4 and given in table 5.3.4 and solve TSP with this matrix and model 2.

Table 5.3.4 Symmetric matrix F for finding the optimal pair in matrix A.
$f_{ij} = (a_{ij} + a_{ji}) : 2$

Node	1	2	3	4	5	6	7	8	Row sum
1	-	20.5	17	17	48	62	58	86	308.5
2		-	11	25	39.5	46.5	48	72	262
3			-	14.5	32	44	42.5	89	250
4				-	41	52.5	47.5	91	288.5
5					-	15.5	12	59.5	247.5
6						-	17.5	48	285.5
7							-	62.5	288
8								-	508

The optimal cycle in the symmetric matrix F turns out to be the same cycle as we found for the optimal cycle in A. The cost in F is 233.5 and then the cost of the optimal pair is 467. Hence, it turns out that in this case the optimal pair is the pair found by solving TSP directly in A.

Example 5.3.3 *An asymmetric matrix decomposed into three different matrices*

Now, let A be divided into the three matrices K, D and E. K is the constant-TSP matrix given by the vectors in example 5.3.1, D is the symmetric matrix in table 5.1.1 and E is given in table 5.3.5 below.

Table 5.3.5 The residual asymmetric matrix E were $A = K+D+E$

Node	1	2	3	4	5	6	7	8
1	-	3	0	0	0	0	3	0
2	0	-	0	0	2	0	0	0
3	4	0	-	1	0	4	2	2
4	0	4	0	-	3	0	0	2
5	1	0	3	0	-	2	0	0
6	0	5	0	3	0	-	0	0
7	0	1	0	0	0	0	-	0
8	2	0	0	0	0	0	0	-

Now, solving TSP for the matrix D gives an optimal solution $\varphi^* = 1-2-8-6-7-5-3-4-1$ with cost 54 as stated above in example 5.1.4. Adding the cost 176 from the constant-TSP given by K, gives a lower bound for TSP in A equal to 230. Now, $\varphi^*(E) = 7$ and $\overleftarrow{\varphi^*}(E) = 0$. This shows that $\overleftarrow{\varphi^*}$ is an optimal solution for A. TSP for A is by this procedure solved as a symmetric problem and the approximation $K+D$ reduces the complexity of this small problem.

Example 5.3.4 *An asymmetric matrix decomposed into a symmetric and a skew-symmetric matrix.*

In theorem 4.2 an asymmetric matrix was decomposed into a symmetric matrix and a skew-symmetric matrix. Using the same matrix A as in table 5.3.5 as an in-put matrix, the decomposition will be the matrix F given in table 5.3.4 and the matrix E given in table 5.3.6 below. Note that this decomposition was also used when the same matrix was transcribed into a Hermitian matrix in the previous section.

Table 5.3.6 The skew-symmetric matrix E constructed by theorem 4.2, based on the matrix A.

Node	1	2	3	4	5	6	7	8
1	-	1.5	-1	1	0	0	2	-1.5
2	-1.5	-	1	-1	1.5	-2.5	0	0
3	1	-1	-	0.5	-2	1.0	0.5	0
4	-1	1	-0.5	-	1	-2.5	-0.5	0
5	0	-1.5	2	-1	-	0.5	0	-0.5
6	0	2.5	-1	2.5	-0.5	-	0.5	0
7	-2	0	-0.5	0.5	0	-0.5	-	-0.5
8	1.5	0	0	0	0.5	0	0.5	-

We have already seen that the optimal cycle φ^* in the matrix F is :

$$1-2-8-6-7-5-3-4-1$$

with cost 233.5. Calculating the cost for this cycle and its revered in the residual matrix E, gives $E(\varphi^*) = 3.5$ and $E(\overleftarrow{\varphi^*}) = -3.5$. Hence, an upper bound for the optimal solution in the original matrix A is 233.5 – 3.5 = 230. This upper bound for an asymmetric matrix is found by solving a symmetric matrix.

Example 5.3.5 *An asymmetric matrix decomposed into a constant-TSP, a symmetric, and a skew-symmetric matrix.*

In theorem 4.3 the decomposition is done in a similar fashion as in theorem 4.2, but some of the asymmetric aspects of the original cost matrix are taken away by introducing a constant-TSP matrix. Repeating example 5.3.4 with an added constant matrix K, we get the following three matrices K, D and E. The matrices are obtained by applying the formulas *(i), (ii),* and *(iii)* in theorem 4.3.

The constant-TSP matrix K is constructed with the help of the two vectors given in table 5.3.7. Note that the cost of any Hamiltonian cycle in this matrix is zero.

Table 5.3.8 The underlying vectors for the matrix K in theorem 4.3

Vector	1	2	3	4	5	6	7	8	Sum
a	0.664	-0.664	0	-0664	-0.133	1.062	-0.794	0.531	0
b	0.312	-0.312	0	-0.312	-0.062	0.5	-0.375	0.250	0

The symmetric matrix turns out to be identical with the symmetric matrix F in table 5.3.4.

The skew-symmetric matrix is given in table 5.3.8.

Table 5.3.8. The skew-symmetric based on theorem 4.3

Node	1	2	3	4	5	6	7	8
1	-	0.875	-1.312	0.375	-0.375	0.187	1.312	-1.062
2	-0.875	-	1.312	-1.000	1.750	-1.687	-0.062	0.562
3	1.312	-1.312	-	0.187	-2.062	1.500	0.125	0.250
4	-0.375	1.000	-0.187	-	1.250	-1.687	-0.562	0.562
5	0.375	-1.750	2.062	-1.250	-	1.062	-0.312	-0.187
6	-0.187	1.687	-1.500	1.687	-1.062	-	-0.375	-0.250
7	-1.312	0.062	-0.125	0.562	0.312	0.375	-	0.125
8	1.062	-0.562	-0.250	-0.562	0.187	0.250	-0.125	-

We have already found the optimal solution to the symmetric matrix in example 5.3.5. Calculating the cost for this cycle in the new skew-symmetric in table 5.3.9, gives 3.5 and hence the reversed cycle has the cost –3.5, which the yields the same result as in example previous example.

Example 5.3.6 *Lower bounds for asymmetric matrices using undirected trees.*

As an example of finding lower bounds for an asymmetric matrix with the help of trees taken from matrices, we will use the asymmetric matrix A in table 5.2.5 and its symmetric component F in table 5.3.4 and the skew-symmetric component E in table 5.3.6.

From the skew-symmetric matrix E we first make a symmetric matrix where all the entries are non-positive, by taking the negative values of the absolute values of the matrix E. The minimal 1-tree in this matrix has the value –14.5 and is illustrated in figure 5.3.1 below.

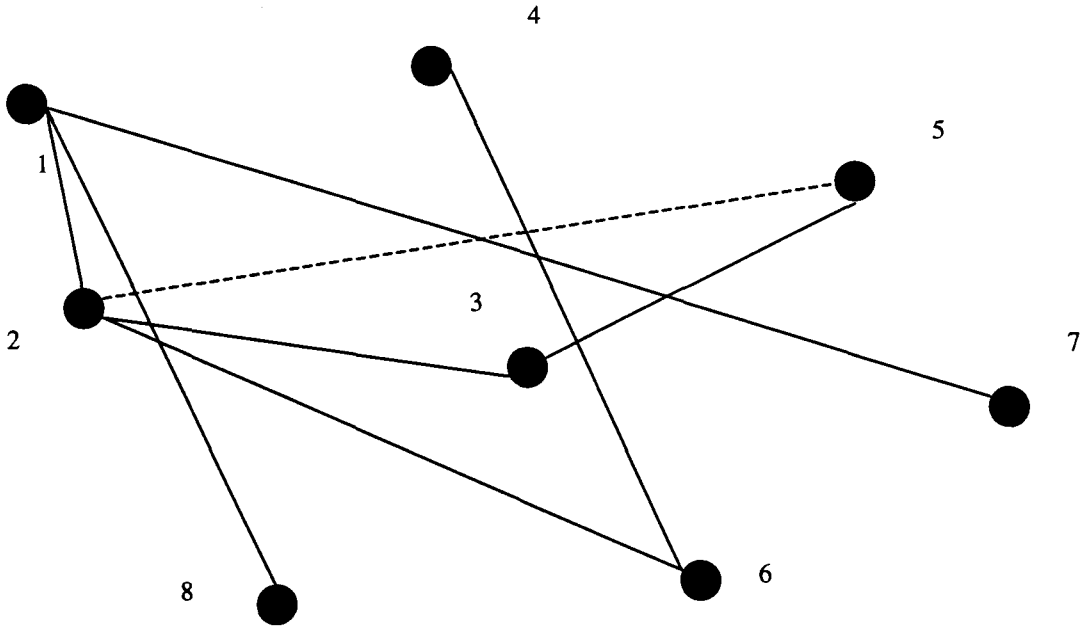


Figure 5.3.1

Then, calculating the maximal spanning trees in each of the saving matrices for the symmetric matrix F, gives the values in the second row in table 5.3.9 The saving matrices can be found in appendix 2.

Table 5.3.9. Lower bounds for TSP for the matrices F and A

	Saving matrix no.							
	1	2	3	4	5	6	7	8
2R(F)	617	525	500	577	495	571	576	1016
MAST(S)	396	297.5	271.5	350.5	266	350	350.5	795
LB(F)	221	226.5	228.5	226.5	229	227	225.5	221
MIST+1	-14.5	-14.5	-14.5	-14.5	-14.5	-14.5	-14.5	-14.5
LB(A)	207	212	214	214	215	213	211	207

The row denoted LB(F) gives different lower bounds for the TSP in the symmetric matrix *F*. The best bound is obtained using saving matrix no. 5. The maximal spanning tree giving the value 266 in this saving matrix is illustrated in figure 5.3.2 below. This lower bound is less than 2% from the optimal solution of 233.5. The last row in the table gives lower bounds for the asymmetric matrix *A*. Since the entries in this matrix are integer numbers, we can use the smallest integer larger than MAST(S) + (MIST+1) as a lower bound for TSP in *A*. The best bound is 215.

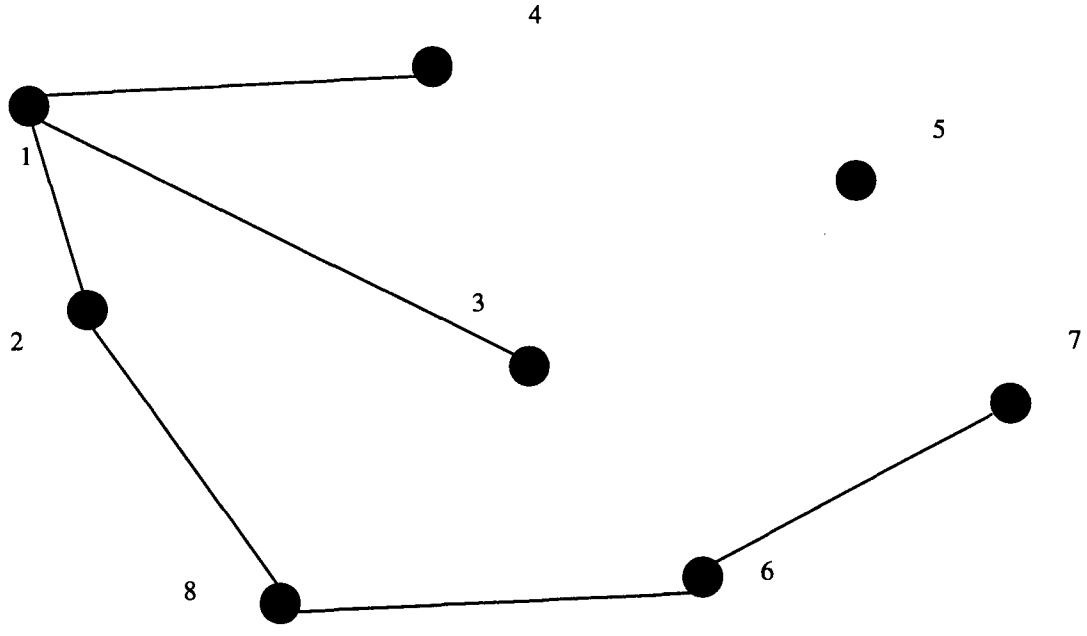


Figure 5.3.2

5.4 ATSP Instances from a Public Library

The following ten ATSP instances can be found at http/testdata/tsp/tsplib/atsp. The names used in the table below are the names used in this library. In addition the number of nodes is given, the optimal solutions, the minimal assignments and the necessary number of sub-tour eliminating constraints applied to reach an optimal solution in each case.

Table 5.4.1 Overview of TSP instances found in public library

Name	Br17	Ftv33	Ftv35	Ftv38	Ftv44	Ftv47	Ftv53	Ry48p	P43	Ftv55
No of nodes	17	34	36	39	45	47	53	48	43	56
Assign ment	0	1185	1381	1438	1521	1652	5931	13827	216	1435
Opt.sol ution	39	1286	1473	1530	1613	1776	6905	14422	5620	1608
No of cons-traints	13	10	31	28	31	48	17	35	47	31

The above TSP instances were all applied to the procedure in chapter 4, that is the matrix was decomposed into a constant-TSP matrix, a symmetric matrix and a residual asymmetric matrix. The original matrix is denoted C, and the other three K, D and E, respectively.

Then the matrix D is solved to optimum in a standard fashion by branch and bound. This gives two results. One will be a lower bound for the TSP for the original matrix found by summing the cost of the TSP in the constant matrix K, plus the cost of the optimal solution of D. This lower bound is denoted $K+D^*$ in the table 5.4.2 below. The second result will be that the procedure above generates an upper bound for the TSP in the original matrix, namely the cost incurred by the optimal Hamiltonian cycle for the symmetric matrix D, denoted φ_D^*, ie , the upper bound will be $C(\varphi_D^*)$.

Table 5.4.2 Upper and lower bounds for TSP for the ten cases.

Name	Br17	Ftv33	Ftv35	Ftv38	Ftv44	Ftv47	Ftv53	Ry48p	P43	Ftv55
$D(\varphi_D^*)$	39	88	56	43	12.5	27	131	1424.75	807	73
No. of sub-cycl.	11	6	12	12	7	11	17	18	44	17
$K+D^*$	39	1260	1399	1483	1489.5	1614	5356	13941.75	955	1425
$C(\psi^*)$	39	1286	1473	1530	1613	1776	6905	14422	5620	1608
Abs. gap	0	26	74	47	123.5	162	1549	480.25	4665	183
$C(\varphi_D^*)$	39	1411	1733	1860	2021	2400	13039	14796	5633	2038
Abs. gap	0	125	203	330	408	624	6134	374	13	415
Dev. %	0	12.0	23.9	25.4	35.7	48.7	143.4	6.1	490	43.0

Table 5.4.2 shows that the decomposition scheme finds the optimal solution in only one out of the ten cases, namely for Br17. The number of sub-tour eliminating constraints used for finding the optimal value for the symmetric matrix D given in the second row, is usually less than the number of such restrictions used when the optimal cycle for C was found, see table 5.4.1. Only for the case P43, the number of restrictions turns out to be the same. In many of the cases there is a substantial decrease in the number of restriction necessary.

The lower bounds found are substantial better than the assignments found by using the original cost matrices. In most of the cases these bounds are rather close to the optimal solutions as can be seen from row no. 5 in table 5.4.2. Again there is one exception, namely the same case as above, P43 and partly that of Ftv53.

The upper bounds found, ie the cost $C(\varphi_D^*)$, are in general not so good. This means that the entries in the asymmetric residual matrices E used in the optimal cycle for D, is in general positive elements and not zeros. There is again one exception where the procedure gives a good upper bound, only 13 above the optimal solution, namely P43.

The last row in table 5.4.2 is calculated as if we do not know the optimal solution and is the standard way of calculating a maximal deviation for a given cycle, when a lower bound is known. The numbers are calculated as follows:

$$\frac{C(\varphi_D^*) - (K + D^*)}{K + D^*}$$

In order to see whether there can be some connections between the structure of the original matrices and the matrices used in the decompositions, the average of the entries and the variations of the entries in the involved matrices are calculated in table 5.4.3.

Table 5.4.3 Average values and variation of the entries in the applied matrices.

Name	Average of the entries				Variation of the entries				var(E):
	C	*K*	*D*	*E*	*C*	*K*	*D*	*E*	var (C)
Br17	14.53	0	14.40	0.13	348	0	340	0.25	0.00071
Ftv33	128.5	34.5	78.5	15.5	3443	467	2424	581	0.17
Ftv35	135.2	37.6	81.4	16.3	3747	517	2626	621	0.17
Ftv38	138.6	38.3	84.4	15.9	3813	483	2687	585	0.15
Ftv44	136.9	32.8	88.8	15.3	3709	353	2867	529	0.14
Ftv47	142.4	33.1	90.1	19.3	3951	370	2925	857	0.22
Ft53	492.9	98.6	201.5	192.9	141419	6690	14292	121799	0.86
Ry48p	1146.6	260.8	841.4	44.4	532096	11175	376719	142596	0.27
P43	593.7	3.44	98,2	492.0	2327353	73	19793	2048215	0.88
Ftv55	131.8	24.1	91.8	15.8	3433	239	2770	600	0.17

From table 5.4.3 one can see that for the first 5 and the last instance, the average value of the entries in the residual matrices *E* are rather small compared to the average value in the original matrices. This means that very much of the costs have been absorbed in the constant matrices and the symmetric matrices in each instance. Hence, finding the optimal cycle in the symmetric matrices *D* and adding the cost of the constant-TSP in the associated matrices *K*, can give good lower bounds for the original cost matrices. This seems to be fairly well illustrated in table 5.4.2. Here, reasonably good lower bounds are found for the same 6 instances. On the other hand, the opposite is found for the two cases Ft53 and P43. In these cases the average values of the entries in the residual matrices *E*, are rather large, as are the variations of the same entries. In these two cases the lower bounds in table 5.4.2 are rather poor. The same effect can be seen taking the ratio between the variation of the entries in *E* and the corresponding variation of the entries in the original cost matrices *C*, that is the numbers in the last column in table 5.4.3. Here, in the first case, these ratios are small (less or equal to 0.17), and the latter case, the ratios are 0.88 and 0.86 respectively. The two last instances in the tables above seem to be some where in between the two extreme cases.

Hence it may be some virtue in calculating the average of the entries in the residual matrix and the variation of these and compare these indicators with the similar numbers for the original matrix. If the indicators are small, then there may be hope that the procedure described in chapter 4 can give good lower bounds. If they are large compared to the similar numbers for the original cost matrix, then there may be little hope of achieving any substantial improvements in the lower bounds.

Further, if the variations of the entries in the residual matrices are rather small, the different costs forthcoming by applying different Hamiltonian cycles will probably

tend to have a relatively low variation as well. This means that the upper bounds found by the procedure will mostly depend on the optimal solution for the symmetric matrix. This effect is to a certain degree reflected in table 5.4.3 by the first five and the last case. In these cases the upper bounds are at least not very far from the optimal solutions.

5.5 Final Comments and Further Research

Final comments

This thesis has mainly been concerned with different decomposition schemes for the underlying matrix for a TSP instance. The different decompositions discussed can be divided into different categories.

One such category will be the linear admissible transformations. Such transformations just adds a constant number to any Hamiltonian cycle and will not in general make it easier to solve a given TSP to optimum. However, such transformations can be utilised in other ways.

One such way is described in chapter 2. Here, a special linear admissible transformation – namely savings - is used to establish that the sum of any Hamiltonian cycle measured in the original cost matrix plus the cost of the same cycle measured in a given saving matrix always is the same. This fact can be utilised in several ways. One such application is to use any of the saving matrices as input to any standard heuristics for TSP. The examples in chapter 2 show that there can be substantial improved in the performance of some of the heuristics when this is done. A second approach is to use the same fact to find simple lower bounds for TSP. These lower bounds are basically found in a traditional way, ie by constructing spanning trees and spanning 1-trees, but now on a saving matrix and finding maximal spanning trees. In combination with the basic relation described in chapter 2, such an approach can give much better lower bounds than can be obtained by the original matrix alone. Further, the new obtained lower bounds can be used in combination with fairly easily solved IP problems – that is assignment problems with one added knap-sack restriction. This can give further improvements and in some cases even solve the original TSP. Finally, the basic relation from chapter 2 can be used to formulate new polynomial classes of TSP. However, these classes are rather small, that is, the extensions from already known classes will not be very substantial, and basically no knew techniques are offered. On the other hand, transforming the original cost matrix into some saving matrix can reveal hidden structures in the original cost matrices.

A second way to apply a linear admissible transformation is in the context suggested by Burkhardt. This approach is used in chapter 4. Here decompositions of the original cost matrix are used to make approximations to the original cost matrix in different ways. Two new concepts are introduced. The first one - called Hamiltonian symmetric matrices. In such matrices, which are easy to identify, it does not matter in which direction the cycle is performed. So, instead of searching for the optimal solution in an asymmetric matrix, it is sufficient to use a symmetric matrix. Hence, in such planning situation the added complexity of an asymmetric cost structure can be avoided. The second concept is the optimal asymmetric pair in asymmetric matrices.

The best pair of a cycle and its reverse can be found, by solving a symmetric TSP. In both these cases, no approximations are taking place.

Another way to decompose an asymmetric matrix is to divide it into three matrices $C = K+D+E$, where K is a constant-TSP matrix, D is a symmetric matrix, and E a residual matrix. Deleting the residual matrix E, one has an approximation for the original matrix C. Whether this will be a good or a bad approximation will depend on the entries in E. If these entries are small or at least do not vary a lot, then we will have a good approximation. The examples in this chapter show that either can be the case. Some indications of what to expect can be done by calculating the average and variation of the entries in E.

In chapter 3 the cost matrix is decomposed into a set of polynomial solvable matrices in the symmetric case. In the asymmetric case one must have a set of skew-symmetric matrices as well. These skew-symmetric matrices are constructed as differences of asymmetric product matrices. The results in chapter 3 indicate that solving asymmetric TSP in general will be more difficult than solving symmetric ones. In the symmetric case, the decomposition can be used to find a lower bound for the TSP instance.

Further research

In chapter 2 some new methods of producing good lower bounds for symmetric TSP were developed. However, it has not been possible to find definite relations between these new bounds or between these new bounds and the more traditional ones. It would have been very convenient to know more about which bounds are the best ones obtainable under given conditions.

Evidently, the new way of finding lower bounds discussed in chapter 2 should be tested on many more and larger TSP-instances than is done in this thesis. In a similar fashion traditional heuristics should be applied to the same TSP-instances using the different saving matrices as input to the heuristics. Further, one should also try how weighted saving matrices behave when used as input to such heuristics. Personally I doubt that using weighted saving matrices to find good lower bounds is a promising path to take. Weighted saving matrices are usually matrices where no row or column will consist of zeros only. This will probably lead to poorer lower bounds. However, introducing weighted saving matrices, give an enormous amount of cost matrices for the same TSP. Now, let a certain heuristic be given. Could one find a set of weights such that this heuristic gives an optimal solution? If so, the problem of finding an optimal solution will be equivalent to finding such a set of weights. Of course it may well be that finding such a set can be as difficult as finding an optimal solution in the first place.

In chapter 2 it was observed that the cost of a Hamiltonian cycle has a precise relation to a specific ellipse. Ellipses have been studied for a very long time and much is known about them and they have a lot of nice properties. It may be possible to utilise these properties in some way in the context of TSP.

In chapter 3, a decomposition of a quadratic and symmetric cost matrix is performed with help of the spectral theorem. The cost matrix can be decomposed in to a certain number of polynomial solvable product matrices.

This decomposition can be used to obtain lower bounds for the original TSP as a side effect we get some upper bounds as well. Further testing is necessary in order to find out whether these bounds are good enough to defend the added computational burden of finding the eigenvalues and eigenvectors.

It will also be interesting to find special structured matrices that yield eigenvectors and eigenvalues that are easy to handle and either give good lower bounds, good upper bounds or even new polynomial classes for TSP. One such example, could be a symmetric cost matrix of some kind yielding eigenvalues that have the same ordering. Such a matrix will then be polynomial solvable.

Asymmetric matrices seem to be more difficult to describe with the help of the spectral theorem. However, a step forward would be to decide whether the skew-symmetric matrices necessary to complete the decomposition for asymmetric matrices are NP-hard or not. If they are, then it seems to strengthen the view that asymmetric matrices in general are more difficult to solve than symmetric ones. If such skew-symmetric matrices turn out to be polynomial solvable, the symmetric and asymmetric matrices are more alike in this respect.

Burkhardts original idea was probably not to have decompositions of a cost matrix yielding equality between the involved matrices, but rather an inequality. Further, he suggests using only matrices that can be solved in polynomial time. In chapter 4, one polynomial class is used but also two which in general will not be polynomial. Formalised Burkhardts idea could be something like the following:

Find two matrices K and P such that K is a constant-TSP matrix and P is polynomial solvable such that

$$K + P \leq C$$

where C is the original cost.

The problem will then be to find suitable candidates for the matrix P.

Further research in this direction will probably yield some rewards.

In the last sub-section in this chapter ten different cases from a public library were tested in context with the content in chapter 4. Similar tests ought to be performed on such large matrices in the context of techniques described in chapter 2. Further, several more matrices ought to be tested as well.

References

Anton, H and Rorres, C	"Elementary Linear Algebra with Applications", John Wiley and Sons, 1987
Baki, Md. Fazle and Kabadi, Santosh	"Pyramidal traveling salesman problem". Computers & Operations Research 26 (1999) pp. 353-369
Balas, Egon	"The Prize Collecting Travelling Salesman Problem". Network, Vol 19 (1989) pp 621-636.
Ball, Magnanti, Monma and Nemhauser (editors)	"Network Models" Handbook in Operations research and management science, Volume 7, North Holland, 1995a
Ball, Magnanti, Monma and Nemhauser (editors)	"Network Routing" Handbook in Operations research and management science, Volume 8, North Holland, 1995b
Bellmore, A. and Hong, S.	"Transformation of multi-salesman problem to the standard traveling salesman problem" J.ACM 21,pp500-504, 1974
Beltrami, E.J. and Bodin, L.D.	"Network and vehicle routing for municipal waste collection". Networks 4, pp 65 - 94,1974
Berenguer; X	" A Characterization of Linear Admissible Transformations for the m-Travelling Salesman problem" European j. Oper. Res. 3, pp 232-249. [4:2], 1979
Biggs, N.L., Lloyd, K. E. and Wilson, R.	"Graph Theory, 1736 - 1936". Clarendon Press, Oxford, 1977
Biggs, N.L.	"T. P. Kirkman, Mathematician" Bull. London Mathematical Soc.,13, pp. 97-120, 1981
Bodin, Golden, Assad and Ball (ed)	"Routing and Scheduling of Vehicles and Crews. The State of the Art" An International Journal Computers & Operations Research. Special Issue, Vol. 10, nr.2 Pergamon Press ,1983
Brey de, M.J.D. and Volgenant, A.	"Well-solved cases of the 2-Peripatetic Salesman Problem" unpublished, paper presented at symposium on combinatorial Optimization (CO96),1996
Burkard, Rainer E.	"Travelling Salesman and assignment Problems: A Survey" Annals of Discrete Mathematics 4 (1979) pp 193-215

Burkard, Rainer E.	"Special Cases of Travelling Salesman Problems and Heuristics". Acta Mathematicae Applicatae Sinica, Vol. 6, no. 3, pp 273 - 288, 1990
Burkard, R.E. and Sandholzer, W.	"Efficiently solvable special cases of bottle-neck travelling salesman problems". Discrete Applied Mathematics, 32, pp 61-76, 1991
Burkard, R.E. and Van der Veen, J.A.	"Universal conditions for algebraic traveling salesman problems to be efficiently solvable" Optimization 22, pp 787-814, 1991
Burkard, Deineko, van Dal, van der Veen and Woeginger	"Well-Solvable Special Cases of the TSP: A Survey" Karl-Franzens-Universität Graz & Technische Universtät Graz, Bericht Nr.52, 1995
Burkov, V.N. and Rubinshtein, M.I.	" A sufficient condition for existence of a Hamiltonian circuit and a new solvable case of the travelling salesman problem". Large scale systems in information and decision technologies", Volume 4, pp 137 - 148, 1983
Clarke, G and Wright, J.W	Scheduling of Vehicles from a Central Depot to a Number of Delivery Points" Opns. Res. 11, pp 568-581, 1963
Colorni, A., Dorigo, M and Maniezzo, V.	"An Investigation of some Properties of an Ant Algorithm". Proceedings of the Parallel Problem Solving from Nature Conference, Brussels. R. Männer and Manderick (eds), Elsevier Publishing,1992
Current, John and Schilling, David	"The Covering Salesman Problem", Transportation Science, Vol.23, No. 3, !989, pp 208-213
Davies, Philip J.	"Circulant Matrices" John Wiley & Sons, 1979
Deineko, V.G, van Dahl, R and Rote, G.	"The convex-hull-and-line Traveling Salesman Problem; A solvable case", Information Processing Letters 51, pp 141 - 149, 1994
Deineko, V.G. and Woeginger, G.J.	"A solvable case of the quadratic assignment problem". Operations Research Letters 22 (1998), pp 13-17.
Desrochers, M and Laporte, G	"Improvements and extensions of the Miller-Tucker-Zemlin subtour eliminating constraints". Operations Research Letters 10, pp. 27 – 36, 1991
Edmonds, J and Johnson, E.L.	"Matching, Euler Tours and the Chinese Postman", Mathematical Programming 5, pp 88 - 124, 1973

Eilon, S; Watson-Gandy, C. and Christofides, N.	"Distribution Management: Mathematical Modelling and Practical Analysis". Hafner, New York, 1971
Finke, G., Burkard, R.E., and Rendl, F.	"Quadratic Assignment Problems". Annals of Discrete Mathematics 31, pp 61-82,1987
Fisher, M.L. and Jaikumar, R.	"A Generalized Assignment Heuristic for Vehicle Routing". Networks, vol. 11, pp. 109 - 124, 1981
Flood, M.M.	"The Traveling-salesman Problem" Oper. res. 4, pp 61 - 75, 1956
Gabovich, E. Ya	"The Small Travelling Salesman Problem" (in Russian) Trudy Vychisl. Tsentra Tartu. Gos. Univ. 19,pp 27-51, 1970
Gaikov, N-E.	"On the minimization of linear form on cycles" (in Russian) Vestsi Akad. Navuk BSSR Ser. fiz.-Mat Navuk 4, p.128, 1980
Garfinkel,R.	"Minimizing Wallpaper Waste, Part 1: A Class of Traveling Salesman Problems" Operations Research, 25 pp 741-751,1977
Gaskell, T.J.	"Bases for vehicle fleet scheduling", Operations research Quarterly, 18 (1967) 281
Gendreau, M. Laporte, G. and Semet, F.	" The covering tour problem". CRT. - 95 -08. Centre de Recherche sur les transports., Université Montréal, 1995
Golden, B., Levy, L. and Dahl, R.	"Two generalizations of the Traveling Salesman Problem". Omega, vol. 9, no.4, pp. 439-441, 1981
Golden, B.L. and Assad, A.A. (ed)	"Vehicle Routing: Methods and Studies" , North Holland, 1988
Golden, B.L., Magnanti, T.L. and Nguyen, H.Q.	"Implementing vehicle routing algorithms", Network, 7, pp 113-148, 1977
Graham, Alexander	"Nonnegative matrices and applicable topics in linear algebra" John Wiley & Sons, 1987
Hadley, S.W., Rendl, F. and Wolkowicz, H	"A New Lower Bound via Elimination for the Quadratic assignment problem", Research Report CORR 89-5, Faculty of Mathematics, University of Waterloo, 1989

Hadley, S.W., Rendl, F. and Wolkowicz, H	"A new lower bound via projection for the quadratic assignment problem". Mathematics of Operations Research, Vol. 17, No. 3 (1992), pp. 727-739.
Hardy, G., Littlewood, JE. and Pólya, G	"Inequalities", Cambridge University Press, (first ed. 1934), second edition 1952, reprinted 1991
Harel, David	"Algorithmics, the Spirit of Computing" Addison-Wesley, 1988
Hart, J.P. and Shogan, A.W.	"Semi-greedy Heuristics: An Empirical Study" Operations research letters, Vol. 6, no.3, pp 107-114, 1987
Halskau,Ø and Jörnsten, K	" The Clark and Wright Heuristic Revisited" Nordic Operations Research Conference, 1995, NOAS'95
Held, M. and Karp, R.M.	"The travelling-salesman problem and minimum spanning trees." Operations Research 18, pp 1138-1162, 1970
Held, M. and Karp, R.M.	"The travelling-salesman problem and minimum spanning trees: Part II" Mathematical Programming 1, pp 6-25, 1971
Jonker, R. and Volgenant, T.	" Identification of Non-optimal Arcs for the Traveling Salesman Problem". Oper. res. letters. vol.1 no. 3, pp 85-88, 1982
Jünger, M., Reinelt, G. and Rinaldi G.	"The Traveling Salesman Problem". Annotated Bibliographies in Combinatorial Optimization, John Wiley & Sons, 1997, Chaper 13.
Kabadi, Santosh and Baki, Md. Fazle	"Gilmore-Gomory type traveling salesman problem". Computers & Operations Research 26 (1999) pp. 329-351
Koopmans, T.C and Beckman, M	"Assignment problems and the location of economic activities". Econometrica, 25, pp53 -76, 1957
Krarup, Jakob	"The Peripatetic Salesman and some related unsolved problems", in Combinatorial programming: methods and applications, ed B. Roy; Reidel Publishing company, Dordrecht, 1975,pp 173 - 178
Laporte, Gilbert	"The Traveling Salesman Problem: An overview of exact and approximate algorithms". European Journal of Operational Research, 59, pp. 231-247,1992

Laporte, Gilbert and Martello, Silvano	"The selective travelling salesman problem". Discrete Applied Mathematics 26, 1990, pp 193-207
Lawler, Eugene	"A solvable case of the traveling salesman problem" Math. Programming1, pp 267-269, 1971
Lawler, Eugene	"Combinatorial Optimization. Networks and Matroids". Holt, Rinehart and Winston, 1976
Lawler, Lenstra et alt(ed)	"The Traveling Salesman Problem. A Guided Tour of Combinatorial Optimization", John Wiley & Sons, 1985
Lenstra, J. K. and Rinnoy Kan, A.H.G.	"A Characterization of Linear Admissible transformations for the m-Travelling Saelesman Problem: A Result of Berenguer" European Journal Oper. Res. 3, pp 250 - 252, 1979
Lin, S.	"Computer solutions for the traveling salesman problem". Bell System Tech. J., 44, pp 2245 - 2269, 1965
Lin, S. and Kernighan, B.W.	"An effective heuristic algorithm for the traveling salesman problem." Oper. res.21, pp 498 - 516, 1973
Little, Murty, Sweeny and Karel	" An algorithm for the travelling salesman problem" Operations Res. 11 (1963) 979
Miller, C.E., Tucker, A.W. and Zemlin, R.A.	"Integer Programming Formulation of Traveling Salesman problems". Journal of the Association for Computing machinery, 7, pp. 326-329, 1960
Nemhauser, G and Wolsey, L	"Integer and Combinatorial Optimization". John Wiley & Sons, 1988.
Norback, J. P. and Love, R. F.	"Geometric approaches to solving the traveling salesman problem". Management Sci. 23,pp 1208 - 1223, 1977
Norback, J. P. and Love, R. F.	"Heuristic for Hamiltonian Path problem in Euclidian twospace". J. Oper, Res. Soc. 30, pp 363 - 368, 1979
Noschang, Mark H.	The Traveling Salesman Problem – a Review of Theory and Current research". http://www.ececs.uc.edu/-mnoschan/sale.html
Or, I.	"Traveling Salesman-Type Combinatorial Problems and their Relation to the Logistics of Regional Blood Banking" Ph.D. thesis, Northwestern University, Evanston, IL.

Orloff, C.	"Routing a fleet of M vehicles to/from a central facility", Networks 4pp 147-162, 1974
Paessens, H.	"The saving algorithm for the vehicle routing problem". European J.of Op.res., 34, pp 336-344, 1988
Papadimitriou, C.H. and Steiglitz, K	"Combinatorial Optimization, Algorithms and Complexity", Prentice-Hall, 1982
Pearn, W.L.	"Solvable cases of the k-person Chinese postman problem". Operations Research letters, 16, pp 242 - 244, 1994
Ramesh, R. and Brown, K.	" An efficient four-phase heuristic for the generalized orienteering problem". Computers Ops Res., Vol. 18, no. 2 pp. 151-165, 1991
Reinelt, Gerhardt	"The Traveling Salesman. Computational Solutions for TSP Applications" Springer Verlag, 1994
Revelle, Charles S. and Laporte, Gilbert	"The plant location problem: New models and research prospects". OR Chronicle, Operations Research, Vol. 44 No. 6 (1996), pp 864-874
Rote, Günther	"The N-line Traveling Salesman Problem". Networks, vol.22 pp. 91 - 108, 1992
Sarvanov, V. I.	"On complexity of minimising a linear form on a set of cyclic permutations" (in Russian) Dokl. Akad. Nauk SSSR 253, pp 533-534, 1980. English translation Soviet Math. dokl. 22, pp 118-120, 1980
Skiena, Steven S.	"The Algorithm Design Manual", Springer, 1998
Stewart jr., W. R.	"A computationally efficient heuristic for the traveling salesman problem" Proc. 13th Annual Meeting of S.E. TIMS, pp75 - 85, 1977
Svetska, J and Huckfeldt, V	"Computational experience with a M-salesmen traveling salesman algorithm", Management Sci. 19, pp 790-799, 1973
Van der Cruyssen, P and Rijkaert, M. J.	"Heuristic for the asymmetric travelling salesman problem". J. Oper. res. Soc. 29, pp 697 - 701, 1978

Van der Veen, J.A.A: "An O(n) algorithm to solve the Bottleneck traveling Salesman Problem restricted to ordered product matrices". Discrete Applied Mathematics 47, pp 57 - 75 , 1993

Vo-Khac "La régularisation dans les problèmes combinatoires et son application dans les problèmes de tourneés." rev. Francaise Automat. Informa. Recherche Opérationelle 2 (1971) 59

Warren, Richard H. "Classes of Matrices for the Traveling Salesman Problem". Linear Algebra and its Applications, 139, pp. 53 - 62,1990

Warren, Richard H. "Optimal Arcs for the Traveling Salesman Problem" Appl. Math. Letters, Vol.5, no. 3 pp. 13-14, 1992

Warren, Richard H. "Special Cases of the Traveling Salesman Problem". Applied Mathematics and Computation, 60, pp. 171-177, 1994

Webb, M. H. J. "Some methods of producing approximate solutions to travelling salesman problems with hundreds or thousands of cities." Oper. res. Quart. 22, pp 49 - 66, 1971

Wong, Richard T. "Integer Programming Formulations of the Traveling Salesman Problem". Proc of the IEEE international conference on Circuits and Computers. Pp. 149-152, 1980

Yelow, P. "A Computational Modification to the Savings Methods of Vehicle scheduling" Operational Research Quarterly, vol. 21, p. 281, 1970

Appendix 1

The saving matrices for the symmetric cost matrix in table 5.1.1.

Saving matrix based on depot in node 1

Node	1	2	3	4	5	6	7	8
1	-	0	0	0	0	0	0	0
2		-	15	5	20	29	20	24
3			-	10	24	27	22	4
4				-	17	20	18	4
5					-	87	84	65
6						-	93	91
7							-	71
8								-

Saving matrix based on depot in node 2

Node	1	2	3	4	5	6	7	8
1	-	0	-5	5	-10	-19	-10	-14
2		-	0	0	0	0	0	0
3			-	0	-1	-7	-3	-25
4				-	2	-4	3	-15
5					-	48	54	31
6						-	54	48
7							-	37
8								-

Saving matrix based on depot in node 3

Node	1	2	3	4	5	6	7	8
1	-	5	0	10	-4	-7	-2	16
2		-	0	0	1	7	3	25
3			-	0	0	0	0	0
4				-	3	3	6	10
5					-	56	58	57
6						-	64	80
7							-	65
8								-

Saving matrix based on depot in node 4

Node	1	2	3	4	5	6	7	8
1	-	-5	-10	0	-17	-20	-18	-4
2		-	0	0	-2	4	-3	15
3			-	0	-3	-3	-6	-10
4				-	0	0	0	0
5					-	50	49	44
6						-	55	67
7							-	49
8								-

Saving matrix based on depot in node 5

Node	1	2	3	4	5	6	7	8
1	-	64	60	67	0	-3	0	19
2		-	55	52	0	6	0	23
3			-	53	0	0	-2	-1
4				-	0	0	1	6
5					-	0	0	0
6						-	6	23
7							-	6
8								-

Saving matrix based on depot in node 6

Node	1	2	3	4	5	6	7	8
1	-	83	85	92	25	0	19	21
2		-	71	68	16	0	10	16
3			-	75	22	0	14	-2
4				-	22	0	17	5
5					-	0	16	-1
6						-	0	0
7							-	-1
8								-

Saving matrix based on depot in node 7

Node	1	2	3	4	5	6	7	8
1	-	64	62	66	0	-9	0	13
2		-	57	51	0	0	0	17
3			-	54	2	-4	0	-5
4				-	-1	-7	0	-1
5					-	-6	0	-6
6						-	0	11
7							-	0
8								-

Saving matrix based on depot in node 8

Node	1	2	3	4	5	6	7	8
1	-	46	66	66	5	-21	-1	0
2		-	57	47	1	-16	-5	0
3			-	72	25	2	17	0
4				-	18	-5	13	0
5					-	1	18	0
6						-	1	0
7							-	0
8								-

Appendix 2

The saving matrices for the symmetric cost matrix in table 5.3.4

Saving matrix based on depot in node 1

Node	1	2	3	4	5	6	7	8
1	-	0	0	0	0	0	0	0
2		-	26.5	12.5	29	36.5	30.5	34.5
3			-	19.5	33	35	32.5	14
4				-	24	26.5	27.5	12
5					-	94.5	94	74.5
6						-	102.5	100
7							-	81.5
8								-

Saving matrix based on depot in node 2

Node	1	2	3	4	5	6	7	8
1	-	0	14.5	28.5	12	4.5	10.5	6.5
2		-	0	0	0	0	0	0
3			-	21.5	18.5	13	16.5	-6.5
4				-	23.5	18.5	25.5	6
5					-	70	75.5	52
6						-	76.5	70
7							-	57.5
8								-

Saving matrix based on depot in node 3

Node	1	2	3	4	5	6	7	8
1	-	7.5	0	14.5	1	-1	1.5	7.5
2		-	0	0.5	3.5	9	5.5	28
3			-	0	0	0	0	0
4				-	5.5	6	9.5	12.5
5					-	60.5	62.5	61.5
6						-	69	85
7							-	69
8								-

Saving matrix based on depot in node 4

Node	1	2	3	4	5	6	7	8
1	-	21.5	14.5	0	10	7.5	6.5	22
2		-	28.5	0	26	31.5	24.5	44
3			-	0	23.5	23	19.5	16.5
4				-	0	0	0	0
5					-	78	76.5	72.5
6						-	82.5	95.5
7							-	76
8								-

Saving matrix based on depot in node 5

Node	1	2	3	4	5	6	7	8
1	-	67	63	72	0	1.5	2	21.5
2		-	60.5	55.5	0	9	3.5	27
3			-	58.5	0	3.5	1.5	2.5
4				-	0	4	5.5	9.5
5					-	0	0	0
6						-	10	27
7							-	9
8								-

Saving matrix based on depot in node 6

Node	1	2	3	4	5	6	7	8
1	-	87.5	89	97.5	29.5	0	21.5	24
2		-	79	73.5	22	0	15.5	22
3			-	82	27.5	0	19	3
4				-	27	0	22.5	9.5
5					-	0	21	4
6						-	0	0
7							-	3
8								-

Saving matrix based on depot in node 7

Node	1	2	3	4	5	6	7	8
1	-	85.5	83.5	88.5	22	13.5	0	34.5
2		-	79.5	70.5	20.5	19.5	0	38.5
3			-	75.5	22.5	16	0	16
4				-	18.5	12.5	0	19
5					-	14	0	15
6						-	0	32
7							-	0
8								-

Saving matrix based on depot in node 8

Node	1	2	3	4	5	6	7	8
1	-	137.5	158	160	97.5	72	90.5	0
2		-	150	138	92	74	86.5	0
3			-	165.5	116.5	93	109	0
4				-	109.5	86.5	106.5	0
5					-	92	110	0
6						-	93	0
7							-	0
8								-