NHH

INSTITUTT FOR FORETAKSØKONOMI

DEPARTMENT OF FINANCE AND MANAGEMENT SCIENCE

FOR 20 2007

ISSN: 1500-4066

AUGUST 2007

Discussion paper

Searching for optimal integer solutions to set partitioning problems using column generation

BY

DAVID BREDSTRÖM, KURT JÖRNSTEN AND **MIKAEL RÖNNQVIST**

Norges Handelshøyskole

NORWEGIAN SCHOOL OF ECONOMICS AND BUSINESS ADMINISTRATION

Searching for optimal integer solutions to set partitioning problems using column generation

David Bredström, Kurt Jörnsten, Mikael Rönnqvist
Department of Finance and Management Science
Norwegian School of Economics and Business Administration
Bergen, Norway
david.bredstrom@gmail.com, kurt.jornsten@nhh.no, mikael.ronnqvist@nhh.no

We describe a new approach to produce integer feasible columns to a set partitioning problem directly in solving the linear programming (LP) relaxation using column generation. Traditionally, column generation is aimed to solve the LP relaxation as quick as possible without any concern of the integer properties of the columns formed. In our approach we aim to generate the columns forming the optimal integer solution while simultaneously solving the LP relaxation. By this we can remove column generation in the branch and bound search. The basis is a subgradient technique applied to a Lagrangian dual formulation of the set partitioning problem extended with an additional surrogate constraint. This extra constraint is not relaxed and is used to better control the subgradient evaluations. The column generation is then directed, via the multipliers, to construct columns that form feasible integer solutions. Computational experiments show that we can generate the optimal integer columns in a large set of well known test problems as compared to both standard and stabilized column generation and simultaneously keep the number of columns smaller than standard column generation.

1. Introduction

There are many important industrial applications that can be formulated as a Set Partitioning Problem (SPP). This is an Integer Programming (IP) problem with binary variables. The standard SPP model can be formulated as

$$\begin{array}{lll} \text{[SPP]} & z^* = \min \displaystyle\sum_{j=1}^{n} c_j x_j & \\ & \text{s.t.} \quad \displaystyle\sum_{j=1}^{n} a_{ij} x_j = 1, & i = 1, \dots, m \\ & \qquad\qquad x_j \in \{0,1\}, & j = 1, \dots, n \end{array}$$

The problem [SPP] is to partition m elements into a number of subsets. Each binary variable or column x_j represents a subset of elements defined through the coefficients a_{ij}. It is well known that to find a feasible solution to SPP is an NP-hard problem in the strong sense. In practice the elements can represent e.g. customers that needs to be visited in a vehicle routing problem. Then each column is a route that visit a subset of the customers with a given cost c_j. Typically, the number of columns, n, is very large and column generation must be used. A general reference on column generation is the book edited by Desaulniers et al. (2005).

A widely used method to solve large scale set-partitioning models is to use Branch and Price (B&P), see e.g. Barnhart et al. (1998), Vanderbeck (1994) or Nemhauser and Wolsey (1988). In this approach the Linear Programming (LP) relaxation of [SPP] is solved using column generation. The structure of the column generation subproblem is dependent on the structure of the underlying application. This subproblem uses the dual information obtained from a master problem where all generated columns (variables) are used. Once the LP-relaxation is solved, a branching strategy is used where additional columns (variables) are added to the master problem in the Branch and Bound (B&B) tree.

In industrial applications it is important to keep the the computational time to find an optimal (or near optimal) solution to SPP as low as possible. There are two traditional approaches (often combined) to improve the performance of the B&P. The first approach is to solve the LP-relaxation (and corresponding LP problems in the B&B tree) as fast as possible. There are several reports to describe how this can be done by using stabilized column generation in where the values of the dual variables are controlled. The idea is to stop the values from oscillating between iterations, which is considered very bad for convergence. Most stabilized column generation algorithms for practical problems found in the literature can be seen as applications of the algorithm found in Amor and Desrosiers (2006), which is a generalization of the methods presented in the earlier work of du Merle et al. (1999). These are proximal point algorithms, where the dual solution is increasingly penalized with respect to the distance from a given point.

The second main approach is to develop efficient branching strategies in order to limit the search tree in the B&B method. There are several reported methods and the more efficient is based on constraint branching, see for example Ryan and Foster (1981) and Barnhart et al. (1998). The overall solution time for the linear master problem can be further reduced with constraint aggregation techniques, such as the algorithm found in Elhallaoui et al. (2005).

There are also other approaches that are used to improve the performance. In order to speed up the time to find feasible solutions, a natural approach is to apply an integer heuristic in order to find a feasible integer solution. Pioneering work was done by Appelgren (1969). A second approach is to apply IP solver only on the columns generated. In this way a full B&P (which can be quite sophisticated and difficult to develop) does not need to be implemented. However, a general view is that this approach often does not generate any feasible integer solution given the columns generated in solving the LP relaxation. The fundamental reason for this is that there is no guarantee that the columns generated are enough, or even useful, for finding an integer solution. This has been noted in for example Hoffman and Padberg (1993), Lübbecke and Desrosiers (2005) and Vanderbeck (1994) and is a well-known phenomenon.

It is sometimes claimed that basic solutions to the LP relaxation are not suitable for column generation due to the fact that they are located in extreme points of the optimal face of the LP polyhedron, See Lübbecke and Desrosiers (2005). To overcome this problem, and instead produce solutions in the relative interior of the optimal face, methods such as analytic center, see Elhedhli and Goffin (2004), bundle methods Briant et al. (2007), and interior point stabilization, Rousseau et al. (2007), have been developed. To date, these methods have not been used extensively for practical problems.

An alternative to solve the LP-relaxation is to make a Lagrangian relaxation in which all (or a subset) of the constraints are relaxed. Here the Lagrangian dual problem can be solved using e.g. a subgradient method, see e.g. Bazaraa et al. (1993). The generation of columns can be done in a similar way as before with the differences that the Lagrangian multipliers are used instead of the dual variables. In order to find a IP solution this approach is combined with a B&B strategy or/and a heuristic approach.

In this paper we describe a new approach that produces integer feasible columns directly in the column generation phase of solving the LP relaxation. The main aim is to generate the columns forming the optimal integer solution. We want to use standard subproblem solvers and hence we need to change the way the values of the dual variables are generated. We base the methodology on a Lagrangian dual formulation where we have introduced one additional surrogate constraint in [SPP] (an aggregation of all other constraints). This constraint is not relaxed in the dual formulation. We also relax the 0/1 restriction on the variables. We use a heuristic multiplier adjustment method which is based on a subgradient method. A subgradient express how well the current multiplier solution match the set partitioning constraint i.e. amount of under and over coverage. In the developed approach only one column can be used in the Lagrangian subproblem. This contrasts a standard dual approach where many columns may be used. It does however provides the possibility to better control the Lagrangian multipliers in order to find columns that match the under coverage. With the additional surrogate constraint we also ensure that we have a dual feasible solution in the current column generation iterate, which is necessary for many pricing problems not to return an already generated column.

The structure of this paper is as follows. In Section 2 we discuss the underlying models and their characteristics used in standard B&P and a standard Lagrangian relaxation. In Section 3 we describe the models used in the proposed method and in Section 4 we outline the proposed algorithm which we call Integer Quality (IQ) column generation. In Section 5 we illustrate and compare some characteristics between the standard, the stabilized and the proposed IQ column generation. Computational experiments and comments are given in Section 6. These tests show that we can generate the optimal integer columns in a large set of well known test problems as compared to both standard and stabilized column generation. In Section 7 we make some conclusions.

2. Models

2.1. Set partitioning problem and stabilized column generation

We can use the sets $J = \{1, \ldots, n\}$ and $I = \{1, \ldots, m\}$ to represent all columns and elements respectively. The LP relaxation of [SPP] then becomes

$$\begin{aligned} \text{[LP]} \qquad z_{LP} = \min & \sum_{j \in J} c_j x_j \\ \text{s.t.} \quad & \sum_{j \in J} a_{ij} x_j = 1, \quad \forall i \in I \\ & x_j \geq 0, \quad \forall j \in J \end{aligned}$$

The dual formulation of [LP] with dual variables u_i is given as

$$\begin{aligned} \text{[LP-dual]} \qquad v_{LP} = \max & \sum_{i \in I} u_i \\ \text{s.t.} \quad & \sum_{i \in I} a_{ij} u_i \leq c_j, \quad \forall j \in J \end{aligned}$$

We assume that problem [LP] is large scale i.e. it has many columns and is solved using column generation. The principle of column generation to solve problem [LP] is illustrated in Figure 1. First an initial set of columns is chosen. Then a Master problem is solved using a limited set of columns. The dual solution for the Master problem is used to define a Subproblem. The purpose of the Subproblem is to identify the least reduced cost column not generated. If the Subproblem finds a new column with reduced cost < 0 it is added to the set of columns. The Master problem and Subproblem are iteratively resolved until no more columns with reduced cost less than 0 is generated.

Set partitioning models are often very degenerate and hence there are many dual solutions. If a standard Simplex method is used to solve [LP] the dual solutions may oscillate between iterations and this is considered bad for convergence. Typically many columns are needed in order to guarantee an optimal solution for problem [LP]. Vanderbeck (1994) has also found that using integer solutions as starting basis solutions for column generation leads to poor performance.

There are several reports to describe how the unwanted oscillation in the dual solution can be avoided by using stabilized column generation. The idea is to stop the dual values from oscillating between iterations by enforcing some limits that can be dynamically changed. Most stabilization

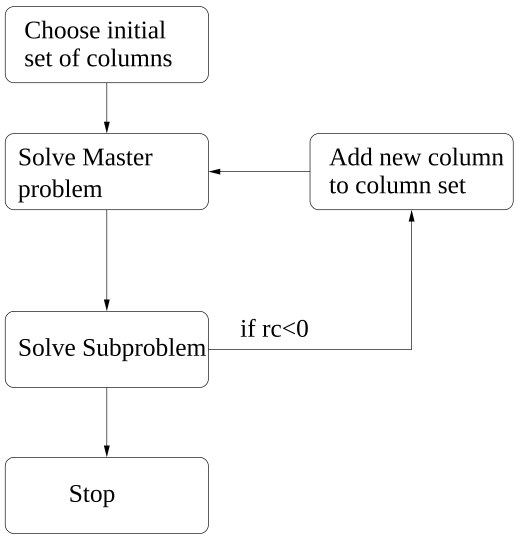

Figure 1 Illustration of column generation.

frameworks have problem dependent parameters that need to be fine tuned for good performance. For the tests presented in this paper, we use the boxstep method of Marsten et al. (1975). In this approach there is only one trust region for the dual variables and we have less parameters to choose, with the potential disadvantage of slower convergence. This model is formulated as

$$\begin{aligned}
\text{[Stab]} \qquad z^s_{LP} = \min & \sum_{j\in J} c_j x_j + \sum_{i\in I}(p_i^+ y_i^+ + p_i^- y_i^-) \\
\text{s.t.} \qquad & \sum_{j\in J} a_{ij}x_j + y_i^+ - y_i^- = 1, \quad \forall i \in I \\
& x_j \geq 0, \quad \forall j \in J \\
& y_i^+, y_i^- \geq 0, \quad \forall i \in I
\end{aligned}$$

In the dual formulation of [Stab] below, the penalty parameters become lower and upper bounds on the dual variables.

$$\begin{aligned}
\text{[Stab-dual]} \qquad v^s_{LP} = \max & \sum_{i\in I} u_i \\
\text{s.t.} \qquad & \sum_{i\in I} a_{ij}u_i \leq c_j, \quad \forall j \in J \\
& -p_i^- \leq u_i \leq p_i^+, \quad \forall i \in I
\end{aligned}$$

We have $z^s_{LP} = z_{LP}$ when $y^+ = y^- = 0$ in an optimal solution. The parameters p_i^+ and p_i^- are initially the average cost per column divided by the number of rows. If y_i^+ or y_i^- are positive when there is no column with negative reduced cost, the corresponding trust region bound, i.e. p_i^+ or p_i^-, is increased by 25%. The convergence criteria for the column generation is when no y-variable is positive and no negative reduced cost columns exists.

2.2. Lagrangian relaxation

The standard Lagrangian relaxation where all partitioning constraints are relaxed is given by

$$\begin{aligned}
\text{[LR]} \qquad \theta(\lambda) = \min & \sum_{j\in J} c_j x_j + \sum_{i\in I} \lambda_i \left(1 - \sum_{j\in J} a_{ij}x_j\right) \\
\text{s.t.} \qquad & x_j \in \{0,1\}, \quad \forall j \in J
\end{aligned}$$

Here, λ_i are the Lagrangian multipliers. These can be interpreted in a similar way as the dual variables to [LP]. This Lagrangian subproblem separates into n one-dimensional problems where the solution is expressed as

$$x_j(\lambda) = \begin{cases} 1 & \text{if } \; c_j - \sum_{i \in I} \lambda_i a_{ij} < 0 \\ 0 & \text{otherwise} \end{cases}$$

The Lagrangian dual problem is to find the best possible value of the dual function $\theta(\lambda)$ and it can be formulated as

$$\text{[LD]} \qquad v_{LD} = \max \, \theta(\lambda)$$

It is well known that we have the property $z_{LP} = v_{LD}$. To solve [LD] a subgradient method can be used, see e.g. Bazaraa et al. (1993). The subgradient $G = (G_1, \ldots, G_m)$ of $\theta(\lambda)$ is given as

$$G_i = 1 - \sum_{j \in J} a_{ij} x_j(\lambda)$$

The value of each component G_i can either be 0, negative or positive indicating how it satisfy, over or under cover the right hand side of the partitioning constraints. The updating of the Lagrangian multipliers in iteration k in a subgradient method follows

$$\lambda_i^{(k)} := \lambda_i^{(k)} + \pi^k G_i$$

where π^k is a step length. Different rules for step length selection can be found in Barahona and Anbil (2002). It is important to note that many columns may be set to 1 in the Lagrangian subproblem and used to compute the subgradient. We have no way to control this number of columns and as a result we may have a large over coverage from a subproblem solution. In the proposed approach we make sure that only one column can be generated and hence we will not have any over coverage. With this property we can better control the values of the Lagrangian multipliers.

3. Models used in the proposed approach

In this section we describe the underlying models that we use in the proposed method. With $m_j = \sum_{i \in I} a_{ij}$ for $j \in J$, the reformulation of SPP with the surrogate constraint is

$$\begin{aligned} \text{[P-IQ]} \qquad z_{IP}^{IQ} = \min \; & \sum_{j \in J} c_j x_j \\ \text{s.t.} \qquad & \sum_{j \in J} a_{ij} x_j = 1, \quad \forall i \in I \\ & \sum_{j \in J} m_j x_j = m \\ & x_j \in \{0, 1\}, \quad \forall j \in J \end{aligned}$$

The LP-relaxation when the restrictions of the binary variables are relaxed of [P] is given as

$$\begin{aligned} \text{[LP-IQ]} \qquad z_{LP}^{IQ} = \min \; & \sum_{j \in J} c_j x_j \\ \text{s.t.} \qquad & \sum_{j \in J} a_{ij} x_j = 1, \quad \forall i \in I \qquad (\gamma) \\ & \sum_{j \in J} m_j x_j = m \qquad\qquad\quad (\delta) \\ & x_j \geq 0, \quad \forall j \in J \end{aligned}$$

The dual formulation of [LP-IQ] is given as

$$
\begin{aligned}
\text{[LP-IQ-dual]} \qquad v_{LP}^{IQ} = \max \; & \sum_{i \in I} (\gamma_i + \delta) \\
\text{s.t.} \qquad & \sum_{i \in I} a_{ij}(\gamma_i + \delta) \le c_j, \quad \forall j \in J
\end{aligned}
$$

The Lagrangian relaxation of [LP-IQ] is

$$
\begin{aligned}
\text{[LR-IQ]} \qquad \phi(\gamma) = \min \; & \sum_{j \in J} c_j x_j + \sum_{i \in I} \gamma_i \left(1 - \sum_{j \in J} a_{ij} x_j \right) \\
\text{s.t.} \qquad & \sum_{j \in J} m_j x_j = m \qquad (\delta) \\
& x_j \ge 0, \quad \forall j \in J
\end{aligned}
$$

We can reformulate [LR-IQ] to

$$
\begin{aligned}
\phi(\gamma) = \sum_{i \in I} \gamma_i + \min \; & \sum_{j \in J} (c_j - \sum_{i \in I} \gamma_i a_{ij}) x_j \\
\text{s.t.} \qquad & \sum_{j \in J} m_j x_j = m \qquad (\delta) \\
& x_j \ge 0, \quad \forall j \in J
\end{aligned}
$$

The dual formulation (with dual variable δ) is

$$
\begin{aligned}
\text{[LR-IQ-Dual]} \qquad \phi(\gamma) = \max \; & \sum_{i \in I} (\gamma_i + \delta) \\
\text{s.t.} \qquad & \sum_{i \in I} (\gamma_i + \delta) a_{ij} \le c_j, \quad \forall j \in J
\end{aligned}
$$

The problem [LR-IQ] has a linear complexity and it is to find the column $\bar{j} \in J$ such that $\bar{j} = \arg\min_{j \in J} (c_j - \sum_{i \in I} a_{ij} \gamma_i)/m_j$, with the optimal solution $x(\gamma)$ such that $x_{\bar{j}}(\gamma) = m/m_{\bar{j}}$ and $x_j = 0$ for $j \in J \setminus \{\bar{j}\}$. The solution to LR-IQ-Dual is $\delta(\gamma) = (c_{\bar{j}} - \sum_{i \in I} a_{i\bar{j}} \gamma_i)/m_{\bar{j}}$. It is clear that the model [LP-IQ-dual] is equivalent to the Lagrangian dual problem

$$
\text{[LD-IQ]} \qquad v_{LD}^{IQ} = \max \; \phi(\gamma)
$$

which is therefore also equivalent to the LP-dual with $u_i = \gamma_i + \delta$ for $i \in I$. For the column generation this indicate that optimized subgradient optimization does not necessarily provide any different columns than those obtained using the dual variables from the LP relaxation.

4. Solution approach

The aim with the proposed Integer Quality (IQ) column generation is to generate columns that define the optimal integer solution. This is achieved by controlling the multiplier values by a heuristic multiplier adjustment method based on sugradient optimization. The column generation will construct columns that fits with other active columns. By active we mean columns that have been used in recent subproblem solutions. An important concept here is that the subproblem solution only have one column active from each iteration. The multiplier update is such that the active columns should not over cover the relaxed constraints. By keeping the number of update iterations low before a new subproblem is solved we can target to generate a column that fits with the part of the constraints which the active columns not cover. A second part of the IQ column generation is to ensure that we have a convergence criteria. This is achieved by making sure that the heuristic multiplier method in the end converges to the standard subgradient method and hence guaranteeing that the correct dual LP solution is found.

Assume that we have a subset $\bar{J} \subset J$ of columns. These columns define the current dual feasible region $\bar{D} = \{u = (u_i) \mid \sum_{i \in I} a_{ij} u_i \leq c_j \ \forall j \in \bar{J}\}$. One requirement for a column generation algorithm to converge is that we obtain a negative reduced cost only if the new column is not represented in the set $\bar{J}$. For many pricing problems it is important not to impose new constraints to achieve convergence, since such constraints typically damage the solvability of the pricing problem. Therefore we need to limit our search for dual prices to the set $\bar{D}$, and a column $\bar{j} \notin \bar{J}$ for which its dual hyperplane intersect with $\bar{D}$.

To construct columns that fits for integer solutions we update the multipliers in a subgradient process by only using a very few iterations. We know that problem [LR-IQ] only have one column active from each iteration. We use the subgradient $G(\gamma)$ defined by $G_i(\gamma) = 1 - \sum_{j \in J} a_{ij} x_j(\gamma)$, where $x(\gamma)$ is the solution to [LR-IQ]. By updating the multipliers few iterations (say up to $p = 5$) according to $\gamma_i^p \leftarrow \gamma_i^{p-1} + \alpha/t G_i(\gamma^{p-1})$, the multipliers will have large values in elements not covered by recently used columns.

In the pricing problem we use $\gamma^p + \delta^p \in \bar{D}$, where $\delta^p = \delta(\gamma^p)$ is the optimal solution to [LR-IQ-Dual] for $\gamma = \gamma^p$. It holds that $\delta^p \geq 0$ for $\gamma^p \in \bar{D}$ and $\delta^p < 0$ otherwise. A column with $\bar{c}_j(\gamma^p) = c_j - \sum_{i \in I} a_{ij} \gamma_i^p < 0$ for $\gamma^p \in \bar{D}$ has a negative reduced cost also with respect to the solution of [LR-IQ-Dual], i.e., $\bar{c}_j(\gamma^p + \delta^p) < 0$. When $\gamma^p \notin \bar{D}$ (and thus $\delta^p < 0$) the sign of the reduced cost is dependent on how well γ_i^p points out a column to fit, and even for columns with $\bar{c}_j(\gamma^p) < 0$ it may hold that $\bar{c}_j(\gamma^p + \delta^p) \geq 0$. With a large steplength α and few iterations we aim for γ^p to be close enough to a perfect match of a wanted column $\bar{j} \notin \bar{J}$, i.e., close to a multiplier vector η such that $\eta_i = \alpha a_{i\bar{j}}$. This column $\bar{j}$ has $\bar{c}_{\bar{j}}(\eta + \delta) < 0$, if its dual hyperplane intersects $\bar{D}$ and if the steplength α is large enough. This is because $\delta = \min_{j \in J}(c_j - \alpha \sum_{i \in I} a_{ij} a_{i\bar{j}})/m_j$ and $m_{\bar{j}} > \sum_{i \in I} a_{ij} a_{i\bar{j}}$ for all column $j \in \bar{J}$.

If we run many iterations the multiplier values will tend to the true optimal dual values. For these values the aim is only to construct columns that best reduce the objective function value without any account to previous columns or integer feasibility.

The proposed IQ column generation is outlined in Algorithm 1, where we use the following parameters

J_0	:	Initial set of columns in [LR-IQ]
T_{init}	:	Initial number of subgradient iterations
T_{max}	:	Maximum number of subgradient iterations
T_{add}	:	Incremental number of subgradient iterations
α	:	Initial step length in subgradient optimization

Algorithm 1 Integer Quality (IQ) column generation

```
Require: $J_0$, $T_{init}$, $T_{max}$, $T_{add}$ and $\alpha$
  $\gamma \leftarrow 0$  //Initial multipliers
  $p \leftarrow 0$  //Iteration counter
  $RC = -\infty$  //Best reduced cost
  $T = T_{init}$
  while $T < T_{max}$ or $RC < 0$ do
    Solve $[\text{LR-IQ}]_p(\gamma)$ (gives $\delta_p$)

    //Column generation.
    Solve $\bar{c}_{j_0} = \min_{j \in J}\{c_j - \sum_i(\gamma_i + \delta_p)a_{ij}\}$
    if $\bar{c}_{j_0} < 0$ then
      Add column $(c_{j_0}, a_{ij_0})$ and update $J_p \leftarrow J_p \cup \{j_0\}$
    end if

    //Iteration adjustment
    $RC \leftarrow \bar{c}_{j_0}$
    if $RC \geq 0$ then
      $T \leftarrow T + T_{add}$  //No new column, increase tolerance
    else
      $T \leftarrow T_{init}$  //Initial number of iterations
    end if

    //Subgradient optimization
    for $t = 1, \ldots, T$ do
      Solve $[\text{LR-IQ}]_{p+1}(\gamma)$
      for $i = 1, \ldots, I$ do
        $G_i \leftarrow 1 - \sum_{j \in J_{p+1}} a_{ij}x_j(\gamma)$
        $\gamma_i \leftarrow \gamma_i + \alpha/tG_i$
      end for
    end for

    $p \leftarrow p + 1$ //New iteration
  end while
```

5. An illustrative example

We have constructed a simple example with eight constraints and 92 columns. The columns represent all combinations which cover 1-3 constraints. The cost coefficients are randomly generated with values in the interval [0.0,1.0] for each constraint covered. The cost coefficients for the columns is hence in the interval [0.0,3.0].

In tables 1-3 we give the results after using standard column generation, stabilized column generation and IQ column generation. Each column give the columns as they are generated. Row "col-id" gives the column index j, "cost" gives c_j and "frac" the optimal fractional solution. The row "integer" is included if any feasible integer solution can be found with the columns generated.

col-id	35	9	21	25	46	53	27	50	49	75	82	73	66	57	60	62
cost	0,51	0,53	0,60	1,05	0,91	1,45	1,28	2,14	2,32	0,39	0,81	0,46	0,64	0,47	0,48	0,67
frac	0,67	0,33	0,33	0,00	0,00	0,00	0,00	0,00	0,00	1,00	0,00	0,67	0,00	0,33	0,00	0,00
	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1
	1	0	0	1	0	0	1	0	0	0	0	0	1	1	0	0
	0	1	0	1	1	0	0	0	0	0	0	1	0	0	0	0
	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
	0	0	0	0	0	1	1	0	1	1	0	0	1	0	1	0
	1	1	0	0	0	1	0	1	0	0	1	0	0	0	0	0
	0	0	1	1	1	1	0	1	0	0	1	1	0	0	0	1
	1	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0

Table 1 Result when standard column generation is used. The number of iterations needed is 16 and no feasible solution is generated.

col-id	58	35	75	73	21	65	9	57
cost	0,22	0,51	0,39	0,46	0,60	0,45	0,53	0,47
frac	0,00	0,67	1,00	0,67	0,33	0,00	0,33	0,33
	1	0	0	0	1	0	1	1
	0	1	0	0	0	1	0	1
	1	0	0	1	0	0	1	0
	0	0	1	0	0	1	0	0
	0	0	1	0	0	0	0	0
	0	1	0	0	0	0	1	0
	0	0	0	1	1	0	0	0
	0	1	0	0	1	0	0	0

Table 2 Result when stabilized column generation is used. The number of iterations needed is 8 and no feasible solution is generated.

Stabilized column generation results, as expected, in less columns than standard column generation. No feasible solution can be found with these two approaches. The IQ column generation do generate the columns needed to construct the optimal integer solution.

Another test was made where we removed a set of columns in order to make sure that only one integer solution exists. The remaining set of 38 columns has the integer solution defined by the columns 2, 3, 38, and 10. We also changed the cost of one column (column 38) in the optimal integer solution so that the duality gap could be made arbitrarily large. The optimal LP objective function value for the first test is 1.57. The optimal integer objective function value for the modified example with $c_{38} = 0.5$ is 3.00 and with $c_{38} = 10.0$ it is 12.50.

The stabilized column generation quickly solve the LP relaxation but column 38 is not generated. This column is extremely bad from a LP relaxed point of view. In fact, when the first column is generated in the stabilized column generation the dual variables are boxed in in such a way that

col-id	35	58	75	21	73	65	9	60	24	57
cost	0,51	0,22	0,39	0,60	0,46	0,45	0,53	0,48	0,59	0,47
frac	0,67	0,00	1,00	0,33	0,67	0,00	0,33	0,00	0,00	0,33
integer	0	0	1	1	0	0	0	0	1	0
	0	1	0	1	0	0	1	1	0	1
	1	0	0	0	0	1	0	0	1	1
	0	1	0	0	1	0	1	0	1	0
	0	0	1	0	0	1	0	0	0	0
	0	0	1	0	0	0	0	1	0	0
	1	0	0	0	0	0	1	0	1	0
	0	0	0	1	1	0	0	0	0	0
	1	0	0	1	0	0	0	0	0	0

Table 3 Result when IQ column generation is used. The number of iterations needed is 10 and the optimal integer solution is generated.

col-id	1	6	4	30	27	14	2	8	5	3
cost	0,51	0,53	0,60	1,20	1,31	0,99	0,39	1,78	0,47	0,46
frac	0,67	0,33	0,33	0,00	0,00	0,00	1,00	0,00	0,33	0,67
	0	1	1	0	0	1	0	1	1	0
	1	0	0	0	1	0	0	1	1	0
	0	1	0	1	0	1	0	0	0	1
	0	0	0	1	0	0	1	1	0	0
	0	0	0	0	1	1	1	0	0	0
	1	1	0	0	0	0	0	0	0	0
	0	0	1	1	1	0	0	0	0	1
	1	0	1	0	0	0	0	0	0	0

Table 4 Result when standard column generation is used for the modified example. The number of iterations needed is 10 and no feasible solution is generated.

col-id	1	6	4	2	3	5
cost	0,51	0,53	0,60	0,39	0,46	0,47
frac	0,67	0,33	0,33	1,00	0,67	0,33
	0	1	1	0	0	1
	1	0	0	0	0	1
	0	1	0	0	1	0
	0	0	0	1	0	0
	0	0	0	1	0	0
	1	1	0	0	0	0
	0	0	1	0	1	0
	1	0	1	0	0	0

Table 5 Result when stabilized column generation is used for the modified example. The number of iterations needed is 6 and no feasible solution is generated.

it is impossible to generate column 38. Standard column generation also does not generate this column. With IQ column generation we generate column 38 with both selection of column cost. In the second case we needed to increase the parameter value α to 10 from 0.8 in the first case with the result that more columns were generated.

col-id	1	14	17	30	2	3	4	6	27	5	31	19	38	10
cost	0,51	0,99	1,13	1,20	0,39	0,46	0,60	0,53	1,31	0,47	1,01	1,56	0,50	1,65
frac	0,67	0,00	0,00	0,00	1,00	0,67	0,33	0,33	0,00	0,33	0,00	0,00	0,00	0,00
integer	0	0	0	0	1	1	0	0	0	0	0	0	1	1
	0	1	1	0	0	0	1	1	0	1	0	1	0	1
	1	0	0	0	0	0	0	0	1	1	0	0	0	1
	0	1	0	1	0	1	0	1	0	0	1	0	0	0
	0	0	1	1	1	0	0	0	0	0	1	0	0	0
	0	1	0	0	1	0	0	0	1	0	0	1	0	0
	1	0	0	0	0	0	0	1	0	0	0	1	0	1
	0	0	1	1	0	1	1	0	1	0	0	0	0	0
	1	0	0	0	0	0	1	0	0	0	1	0	1	0

Table 6 Result when IQ column generation is used for the modified example and where $c_{38} = 0.50$. The number of iterations needed is 14 and the optimal integer solution is generated.

col-id	1	14	17	30	2	3	20	6	12	4	25	29	27	23	5	38	10	32	26	34	37	31	19	15
cost	0,51	0,99	1,13	1,20	0,39	0,46	1,55	0,53	1,38	0,60	1,58	2,16	1,31	2,45	0,47	10,00	1,65	1,55	1,68	1,26	0,61	1,01	1,56	1,18
frac	0,67	0,00	0,00	0,00	1,00	0,67	0,00	0,33	0,00	0,33	0,00	0,00	0,00	0,00	0,33	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
integer	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0
	0	1	1	0	0	0	1	1	1	1	0	0	0	1	1	0	1	0	0	0	0	0	1	1
	1	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	0	0	0	0
	0	1	0	1	0	1	0	1	0	0	1	1	0	0	0	0	0	1	0	1	0	1	0	1
	0	0	1	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	1	0	0
	0	1	0	0	1	0	1	0	0	0	1	0	1	0	0	0	0	1	0	1	0	0	1	0
	1	0	0	0	0	0	0	1	0	0	0	1	0	1	0	0	1	1	0	0	0	0	1	0
	0	0	1	1	0	1	1	0	0	1	0	0	1	0	0	0	0	0	1	0	1	0	0	0
	1	0	0	0	0	0	0	0	1	1	0	0	0	1	0	1	0	0	0	1	0	1	0	1

Table 7 Result when IQ column generation is used for the modified example and where $c_{38} = 10.0$. The number of iterations needed is 25, and the optimal integer solution is generated.

6. Results

For the experiments we use a standard 2.66 GHz Pentium IV with 2 GB of internal memory. The 40 test problems consists of real-world SPP instances from OR-Library. They originate from the airline crew scheduling problem found in Hoffman and Padberg (1993) and have also been used in e.g. Cavalcante et al. (2006) and Joseph (2002). There are four different types of instances with the labels aa, kl, nw and us. Among the available instances, we considered only those problems having an integrality gap. The instances have different properties when it comes to number of columns needed for an integer solution, average costs of the the columns etc. The settings used for the results presented are given in Table 8.

Label	α	T_{max}	T_{init}	T_{add}
aa	2	200	5	5
kl	0.2	100	5	5
nw	200	100	5	5
us	10	100	5	5

Table 8 Parameter settings in IQ column generation.

In Table 9 we give the results from the experiments. Column "Label" indicate the reference name of the test problem in the OR-Library, "m" the number of constraints, "n" the number of columns, "LP" is the LP relaxation value, "Opt" the optimal integer objective function value, "Value" the best integer objective function value among the generated columns for each of the three tested methods (IQ column generation, Standard column generation and Stabilized column generation) and "Columns" the number of columns generated in each of the methods. The last line gives the total number of columns generated for all problems. Values indicated in boldface reflect that the optimal function value is generated and "-" indicate that no feasible integer solution is found.

Label	Dimensions		LP	Opt	Value			Columns		
	m	n			IQ	Std	Stab	IQ	Std	Stab
aa01	823	8904	55535.4	56138	56213	56888	-	1877	1937	1774
aa03	825	8627	49616.4	49649	49663	49664	**49649**	1434	2003	1246
aa04	426	7195	25877.6	26374	**26374**	-	-	1006	1026	779
aa05	801	8304	53735.9	53839	**53839**	54087	**53839**	1420	1850	1265
aa06	646	7292	26977.2	27040	27063	27091	-	1556	1499	937
kl01	55	7479	1084	1086	**1086**	1101	1096	140	152	134
kl02	71	36699	215.3	219	**219**	221	-	295	225	152
nw03	59	43749	24447	24492	**24492**	32823	**24492**	315	228	147
nw04	36	87482	16310.67	16862	16956	-	-	460	154	300
nw06	50	6774	7640	7810	**7810**	**7810**	-	189	135	91
nw11	39	8820	116254.5	116256	116265	116265	116265	106	115	96
nw13	51	16043	50132	50146	**50146**	50298	50158	258	170	129
nw17	61	118607	10875.7	11115	**11115**	13998	**11115**	630	254	89
nw18	124	10757	338864.3	340160	**340160**	365090	341282	552	294	555
nw20	22	685	16626	16812	**16812**	-	17634	49	53	38
nw21	25	577	7380	7408	**7408**	7442	-	41	53	23
nw22	23	619	6942	6984	**6984**	7158	7158	49	58	21
nw23	19	711	12317	12534	**12534**	18760	-	51	39	42
nw24	19	1366	5843	6314	**6314**	6396	-	34	58	17
nw25	20	1217	5852	5960	**5960**	7448	-	49	54	32
nw26	23	771	6743	6796	**6796**	6842	6942	41	58	27
nw27	22	1355	9877.5	9933	**9933**	9972	-	30	70	19
nw28	18	1210	8169	8298	**8298**	**8298**	-	17	43	27
nw29	18	2540	4185.9	4274	**4274**	4432	**4274**	94	52	50
nw30	26	2653	3726.8	3942	**3942**	5468	4342	46	64	35
nw31	26	2662	7980	8038	**8038**	8144	**8038**	56	67	52
nw32	19	294	14570	14877	**14877**	-	-	34	34	29
nw33	23	3068	6484	6678	6724	**6678**	6682	35	74	34
nw34	20	899	10453.5	10488	**10488**	**10488**	**10488**	28	58	26
nw35	23	1709	7206	7216	**7216**	10796	-	53	66	19
nw36	20	1783	7260	7314	**7314**	7506	7328	94	67	71
nw37	19	770	9961.5	10068	10524	11244	**10068**	27	45	26
nw38	23	1220	5553	5558	**5558**	5592	**5558**	72	60	36
nw39	25	677	9868.5	10080	**10080**	10512	-	27	56	20
nw40	19	404	10658.3	10809	**10809**	-	11049	24	40	19
nw41	17	197	10972.5	11307	**11307**	12102	-	17	33	15
nw42	23	1079	7485	7656	7666	7666	7666	46	64	31
nw43	18	1072	8897	8904	**8904**	**8904**	**8904**	48	57	33
us01	145	1053137	9963.07	10036	**10036**	10417	-	353	393	292
us04	163	28016	17731.7	17854	**17854**	19128	**17854**	179	359	247
							Total	11832	12117	8975

Table 9 Results on the SPP instances in OR-Library having a positive integrality GAP.

IQ column generation generates feasible solutions in all cases and optimal solutions in 32 out of 40 test problems. Using stabilized column generation feasible solutions were found in 23 cases and optimal in 12. The corresponding numbers for standard column generation is 35 feasible and 6 optimal. Both standard and stabilized column generation fails in several cases to generate columns that form integer feasible solutions, 5 for standard and 17 for stabilized column generation. The number of columns generated in IQ column generation is in between the number of columns generated in stabilized and standard column generation. Table 10 summarizes the number of feasible and optimal integer solutions generated by each of the tested methods.

Characteristic	Standard CG	stabilized CG	IQ CG
# of problems with feasible solutions	35	23	40
Proportion of problems with feasible solutions	87.5%	57.5%	100.0%
# of problems with optimal solutions	5	11	32
Proportion of problems with optimal solutions	12.5%	27.5%	80.0%

Table 10 Summary of the results in table 9.

The proposed IQ column generation uses parameters which works differently for instances. This is no difference than e.g. using stabilized column generation see e.g. du Merle et al. (1999). We can find parameter settings for IQ column generation that works for all test problems. With the same setting it does work in generating the columns needed to define the optimal (or near optimal) integer solution. The drawback is an increase of the number of columns generated. A large value of the parameter α can be viewed as an over generation of columns (as compared to a more adjusted and smaller value for each instance). In experiments not presented we have been able to establish the optimal integer solution for all test problems by changing the value of α for each problem.

Each column defines a plane in the dual space. In the dual optimal solution we can identify which columns that makes up the optimal relaxed LP solution. The columns making up the integer optimal solution may require columns generated by quite different dual solutions. In stabilized column generation the aim is to limit the range of the dual values and reduce the number of columns generated as much as possible. Moreover, as more columns are included the dual space that can be searched is limited by each iteration. If we assume that the columns making up feasible integer solutions require different dual values to be generated it is likely that less feasible integer solutions are generated by stabilized column generation as compared to standard column generation. This is supported in the experiments.

We have not compared the CPU times between the approaches. Stabilized column generation generate fewer columns as compared to standard column generation. However, in some iterations when the y-variables are not 0, new subproblems need to be solved. In IQ column generation, new subproblems need to be solved if no column could be generated because of too few subgradient iterations. The subgradient optimization is easy to implement and not computationally expensive, especially since we aim to use as few subgradient iterations as possible.

7. Conclusions

The proposed IQ column generation is novel and can generate columns that define the optimal (or near optimal) integer solution. The results show that we in all test problems can generate optimal or near optimal solutions. In the tests we could generate columns that define the optimal integer solutions in 32 out of 40 test problems. Feasible solution were found in 40 out of 40 problems. This result contrast the generally accepted property that optimal and near optimal solutions often not can be found in solving the IP problem defined by the columns generated by the LP-relaxation. This has a very positive impact for industrial cases where quick solution times are important. IQ column generation also solves the LP relaxation which is used as a convergence criteria.

The IQ column generation is based on extending a standard SPP model with a surrogate constraint that is not relaxed in a Lagrangian relaxation framework. With this surrogate constraint we obtain the property that only one column is used in the subproblem solution. This has the effect that we better can control how the multiplier values are updated. The proposed multiplier adjustment method is based on a subgradient method where the accuracy of the steplength can be used to balance what type of column we generate. We can choose to aim for columns that strive to find feasible integer solutions or the optimal (LP optimal) solution by choosing different parameter settings.

We have identified three areas interesting for further research. The first is to establish a linkage between generating the optimal integer solutions and the parameter setting of α. The second is to study other structured models where it is hard to find high quality integer solutions. Examples are the generalized assignment problem and the facility location problem. These can also be formulated as SPP but there are other characteristics of the problems that can be explored in order to find good multiplier values. The third is to study how the proposed multiplier updating can be used in other applications using Lagrangian relaxation to search for feasible integer solutions.

References

Amor, H. B., J. Desrosiers. 2006. A proximal trust-region algorithm for column generation stabilization. *Computers and Operations Research* **33** 910–927.

Appelgren, L. H. 1969. A column generation approach for a ship scheduling problem. *Transportation Science* **3** 53–68.

Barahona, F., R. Anbil. 2002. On some difficult linear programs coming from set partitioning. *Discrete Applied Mathematics* **118** 3–11.

Barnhart, C., E.L. Johnson, G.L. Nemhauser, M. W. P. Savelsbergh, H. Vance. 1998. Branch-and-price: Column generation for solving huge integer programs. *Operations Research* **46**(3) 316–332.

Bazaraa, M. S., H. D. Sherali, C. M. Shetty. 1993. *Nonlinear Programming: Theory and Algorithms*. John Wiley & Sons.

Briant, O., C. Lemaréchal, Ph. Meurdesoif, S. Michel, N. Perrot, F. Vanderbeck. 2007. Comparison of bundle and classical column generation. *Mathematical Programming* **doi: 10.1007/s10107-006-0079-z**.

Cavalcante, V. F., C. C. de Souza, A. Lucena. 2006. A relax-and-cut algorithm for the set partitioning problem. *Computers and Operations Research* **doi: 10.1016/j.cor2006.10.009**.

Desaulniers, G., J. Desrosiers, M. M. Solomon. 2005. *Column generation*. Kluwer Academic Publishers.

du Merle, O., D. Villeneuve, J. Desrosiers, P. Hansen. 1999. Stabilized column generation. *Discrete Mathematics* **194** 229–237.

Elhallaoui, I., D. Villeneuve, F. Soumis, G. Desaulniers. 2005. Dynamic aggregation of set-partitioning constraints in column generation. *Operations Research* **53**(4) 632–645.

Elhedhli, S., J.-L. Goffin. 2004. The integration of an interior-point cutting plane method within a branch-and-price algorithm. *Mathematical Programming* **100** 267–294.

Hoffman, K. L., M. Padberg. 1993. Solving airline crew scheduling problems by branch-and-cut. *Management Science* **39**(6) 657–682.

Joseph, A. 2002. A concurrent processing framework for the set partitioning problem. *Computers and Operations Research* **29** 1375–1391.

Lübbecke, M. E., J. Desrosiers. 2005. Selected topics in column generation. *Operations Research* **53**(6) 1007–1023.

Marsten, R. E., W. W. Hogan, J. W. Blankenship. 1975. The boxstep method for large-scale optimization. *Operations Research* **23**(3) 389–405.

Nemhauser, G. L., L. A. Wolsey. 1988. *Integer and combinatorial optimization*. John Wiley & Sons.

Rousseau, L.-M., M. Gendreau, D. Feillet. 2007. Interior point stabilization for column generation. *Operations Research Letters* **doi:10.1016/j.orl.2006.11.004**.

Ryan, D. M., B. A. Foster. 1981. *Computer scheduling of public transport urban passanger vehicle and crew scheduling*, chap. An integer programming approach to scheduling.

Vanderbeck, F. 1994. Decomposition and column generation for integer programs. Ph.D. thesis, Universite' Catholique do Lovain.