Column Generation in the Integral Simplex Method

Elina Rönnberg and Torbjörn Larsson

Original publication:

Elina Rönnberg and Torbjörn Larsson, Column Generation in the Integral Simplex Method, 2009, European Journal of Operational Research, (192), 1, 333-342.
http://dx.doi.org/10.1016/j.ejor.2007.09.037.

Column Generation in the Integral Simplex Method

Elina Rönnberg* and Torbjörn Larsson*

September 20, 2007

Abstract: The integral simplex method for set partitioning problems allows only pivots-on-one to be made, which results in a primal all-integer method. In this technical note we outline how to tailor the column generation principle to this method. Because of the restriction to pivots-on-one, only local optimality can be guaranteed, and to ensure global optimality we consider the use of implicit enumeration.

Keywords: Integer programming, Set partitioning, Column generation, Implicit enumeration, Quasi-integrality.

*Department of Mathematics, Linköping University, SE-581 83 Linköping, Sweden.
Corresponding author: e-mail elron@mai.liu.se

1 Introduction

During the past few decades, the column generation principle has gained considerable popularity for solving various classes of decision problems of practical interest; see for example [12] and [7] for recent surveys of applications and methodology. Column generation has become popular within combinatorial optimization, especially in the fields of routing and scheduling. Since column generation is a pure linear programming method, it then needs to be complemented with appropriate integer programming techniques. For example, if it is combined with branch-and-bound, the branch-and-price method is obtained, see e.g. [3].

In many applications of column generation, the available columns are combined by solving linear programming relaxations of set partitioning master problems. A nice example of this is the generalized assignment problem, see [9]. Set partitioning problems have the interesting quasi-integrality property, first shown by Trubin [11]; this property implies that it is possible to move between two bases that are adjacent but associated with different integer vertices, by making a simplex pivot on a one-entry in the tableau. The *integral simplex method*, which was briefly described in [11], was named by Yemelichev et al. [13], and recently further developed and also applied by Thompson [10], makes use of this fact for solving the set partitioning problem.

We outline how to tailor the column generation principle to the integral simplex method for set partitioning problems, by using a theoretical result derived by Balas and Padberg [1]. Together with implicit enumeration, that assures that an optimal solution is found, we obtain a novel primal all-integer column generation method for combinatorial optimization problems with set partitioning master problems.

The remainder of the paper is organized as follows. Some theoretical background concerning the set partitioning problem is given in Section 1.1. The inclusion of column generation into the integral simplex method, and its combination with implicit enumeration, is made in Section 2. Concluding remarks are made in Section 3.

1.1 Preliminaries

Consider the set partitioning problem

$$
\begin{aligned}
[SPP] \qquad z^* \;=\; & \min \sum_{j\in N} c_j x_j \\
& \text{s.t.} \quad \sum_{j\in N} a_j x_j = e, && \text{(1a)} \\
& \qquad x_j \in \{0,1\}, \quad j \in N, && \text{(1b)}
\end{aligned}
$$

where $N = \{1, \ldots, n\}$ is the set of indices for the variables, $a_j = (a_{1j}, \ldots, a_{ij}, \ldots, a_{mj})^{\mathrm{T}}$, $j \in N$, are vectors of zeros and ones, c_j, $j \in N$, are integers, and $e = (1, \ldots, 1)^{\mathrm{T}}$ is an m-vector. The index set for the constraints is $M = \{1, \ldots, m\}$. We assume that $n \geq m$ and that the problem is feasible. We assume further, without loss of generality, that the matrix $(a_1, \ldots, a_n)$ has no zero row or column (since a zero row would imply infeasibility and a variable associated with a zero column can be eliminated) and full rank.

Let SPP^{LP} be the linear programming relaxation of the problem SPP, obtained by replacing (1b) with $x_j \geq 0$, $j \in N$. For any basic feasible solution to SPP^{LP}, we denote the basis by $B \subseteq \{a_1, \ldots, a_n\}$. Let I and J denote the corresponding sets of basic and non-basic columns, respectively, and let $u^{\mathrm{T}} = c_B^{\mathrm{T}} B^{-1}$, where $c_B = (c_j)_{j\in I}$, be the complementary dual basic solution. Further let $Q \subseteq I$ be the set of basic variables that takes the value one. Two bases are called *adjacent* if they differ in exactly one column.

In the remainder of this section, we will present a background to the column generation method to be proposed.

Definition 1 *Let X be a polytope and X_I its set of integer points. The polytope X is called* quasi-integral *if every edge of the convex hull of X_I is also an edge of X.*

This property implies that any two integer points of the polytope can be joined by a path that consists only of such edges of the polytope that connect integer points. The following result is from Trubin [11]. (He did however not use the term quasi-integral.)

Theorem 1 *The polytope described by the constraints* (1a) *and $x_j \geq 0$, $j \in N$, is quasi-integral.*

See also Yemelichev et al. [13, pp. 189–193] for an introduction to quasi-integrality. Other problems with this property include the simple plant location problem (*ibid.*),

the uncapacitated network design problem, see [4], and the one and two facility, one commodity, network design problems, see [8]. The perfect matching and set packing problems also have this property since they can be stated as set partitioning problems.

The set partitioning problem has the following interesting properties, which are shown by Balas and Padberg [1].

Theorem 2 *Let x^1 be a feasible and integer solution to SPP^{LP}, associated with the basis B_1, and suppose that x^1 is not optimal in SPP. Denote by x^2 an optimal solution to SPP, and let B_2 be an associated basis in SPP^{LP}. Let J_1 and Q_2 be the index sets of the non-basic columns in B_1 and the one-valued basic variables in x^2, respectively. Then there exists a sequence of adjacent bases $B_{10}, B_{11}, B_{12}, \ldots, B_{1p}$ in SPP^{LP}, such that $B_{10} = B_1$, $B_{1p} = B_2$, and (a) the associated vertices $x^1 = x^{10}, x^{11}, x^{12}, \ldots, x^{1p} = x^2$, are all feasible and integer, (b) $cx^{10} \geq cx^{11} \geq \ldots \geq cx^{1p}$, and (c) $p = |J_1 \cap Q_2|$.*

Since any vertex can be made optimal by adjusting the objective function, it follows from the theorem that for *any* two integer feasible basic solutions to SPP^{LP}, x^1 and x^2, there exists a sequence of adjacent bases with length $p = |J_1 \cap Q_2|$, such that the associated vertices are all feasible and integer.

Theorem 2 however is not useful for the practical solution of the set partitioning problem, since it requires knowledge of an optimal solution. As remarked by Balas and Padberg [1], the construction of the sequence of adjacent bases may involve degenerate pivots on negative tableau entries, which is not allowed in the simplex method. Furthermore, the standard anti-cycling rules are not known to work when negative pivots are performed.

As proposed in Balas and Padberg [2], a possible approach to solve the set partitioning problem would be to repeatedly find edges connecting the current vertex to an adjacent integer vertex with a better objective value. The movement along an edge between two integer vertices corresponds algebraically to a non-degenerate simplex pivot on a one entry. Because of degeneracy it can be hard to find a basis that enables such a pivot. The idea of moving between integer vertices by making pivots on one-entries was first suggested in [11], and it is the foundation for the integral simplex method.

For future reference, we make the following definition of the pivots used in the integral

simplex method. Here, $\bar{e} = B^{-1}e$ and $\bar{a}_j = B^{-1}a_j$, $j \in N$, are the updated right-hand-side and constraint columns, respectively, and $\bar{c}_j = c_j - u^{\mathrm{T}}a_j$, $j \in J$, are the reduced costs.

Definition 2 *Given a simplex tableau associated with an integer basic feasible solution to SPP^{LP} and an $s \in J$ such that $\bar{c}_s < 0$.*

(i) A non-degenerate pivot-on-one *is a pivot operation on an entry $\bar{a}_{rs} = 1$ such that*

$$\min_{i \in M} \left\{ \frac{\bar{e}_i}{\bar{a}_{is}} \;\middle|\; \bar{a}_{is} > 0 \right\} = \frac{\bar{e}_r}{\bar{a}_{rs}} = 1.$$

(ii) A degenerate pivot-on-one *is a pivot operation on an entry $\bar{a}_{rs} = 1$ such that*

$$\min_{i \in M} \left\{ \frac{\bar{e}_i}{\bar{a}_{is}} \;\middle|\; \bar{a}_{is} > 0 \right\} = \frac{\bar{e}_r}{\bar{a}_{rs}} = 0.$$

Both (i) and (ii) are referred to as a pivot-on-one.

Only making pivots-on-one in the simplex method may be insufficient to reach an optimum, as illustrated by the following example.

Example 1 Consider the following set partitioning problem, given in [10].

$$\begin{array}{rcll} z^* & = & \min & \begin{pmatrix} 72 & 48 & 77 & 44 & 56 & 49 & 77 & 41 & 47 & 96 & 42 \end{pmatrix} x \\ & & \text{s.t.} & \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} x = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \\ & & & x \in \{0,1\}^{11} \end{array}$$

The following simplex tableau is obtained for $I = \{1, 6, 7, 8, 11\}$.

$x_j \in I$	0	71	-53	-45	-116	0	0	0	28	36	0	-149
x_1	1	-1	2	2	2	0	0	0	-1	-1	0	1
x_6	0	1	-1	-1	-1	1	0	0	1	1	0	0
x_7	0	0	1	1	1	0	1	0	0	0	0	1
x_8	0	0	0	-1	0	0	0	1	0	1	0	0
x_{11}	0	0	-1	-1	0	0	0	0	1	1	1	0

This integer solution is not optimal, but it is not possible to make a pivot-on-one. ■

The situation appearing in the example naturally leads to the following definition.

Definition 3 *An integer basic feasible solution to SPP^{LP} is called a* local optimum *if it is not possible to make a pivot-on-one.*

Note that the local optimality property is associated with the choice of basis at the vertex, and not with the vertex itself. An optimal solution to SPP is henceforth referred to as a *global optimum*, and such a point has clearly been found if $\bar{c}_j \geq 0$, $j \in N$, holds.

A way to escape from a local optimum, with the aim to eventually find a global optimum, is to pivot on a negative tableau entry ($\bar{a}_{rs} < 0$) in a degenerate row ($\bar{e}_r = 0$); cf. the comment after Theorem 2.

Example 1 (continued) By letting variables x_5 and x_6 become basic and non-basic, respectively, the following tableau is obtained.

$x_j \in I$	0	-45	63	71	0	-116	0	0	-88	-80	0	-149
x_1	1	1	0	0	0	2	0	0	1	1	0	1
x_5	0	-1	1	1	1	-1	0	0	-1	-1	0	0
x_7	0	1	0	0	0	1	1	0	1	1	0	1
x_8	0	0	0	-1	0	0	0	1	0	1	0	0
x_{11}	0	0	-1	-1	0	0	0	0	1	1	1	0

It is now possible to make a pivot-on-one and even a non-degenerate pivot-on-one, by letting x_2 become basic. ■

Another strategy for handling the existence of local optima is to partition the feasible set by applying a branching technique, such as in the *global* integral simplex method (GLISM) developed by Thompson [10]. This method creates a search tree of subproblems, each of which is solved by the *local* integral simplex method (LISM). To create an initial integer basic feasible solution in the LISM, artificial variables with large costs are introduced. Standard techniques for preventing cycling in the simplex method are used, even though they are not guaranteed to work. (GLISM is outlined below, to simplify for the reader.)

The branching technique used in GLISM leads to fixations of variables, and in every subproblem k of the tree the following steps are performed.

1. Let F_k^0 and F_k^1 be the sets of variables that shall be set to zero and one, respectively, and reduce the problem according to the following steps: (i) Let $x_j = 1$, $j \in F_k^1$, and form $w = \sum_{j \in F_k^1} a_j$. (ii) Let $x_j = 0$ if $a_{ij} = w_i = 1$ for some row $i \in M$. (iii) Cross out all rows $i \in M$ with $w_i = 1$. (iv) Let $x_j = 0$, $j \in F_k^0$. (v) Cross out all columns $j \in N$ with $a_{ij} = 0$ for all remaining rows i.

2. If the problem is empty or contains zero-rows, then go to Step 5.

3. Solve the reduced problem by applying the LISM. If all artificial variables equal zero and the solution is the best one found this far, then save the solution. If there are no negative reduced costs, then go to Step 5.

4. Branch and create a new subproblem for every variable with negative reduced cost, as illustrated in Figure 1. Choose a new branch to examine.

5. Cut the branch. If there is no branch left, then terminate, with the best solution saved being optimal. Otherwise examine a new branch.

Thompson [10] applies GLISM to randomly generated instances and problems arising in crew scheduling [5] and his computational experience is promising.

When using LISM, if the number of variables is much larger than the number of constraints (i.e. $n \gg m$), then there are many non-basic variables from among which a new basic variable can be chosen. It is resonable to assume that this would increase the

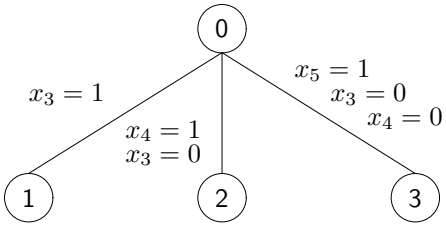

Figure 1: The branching technique in GLISM, as applied to the first tableau in Example 1. Here, the variables with negative reduced costs are x_3, x_4, and x_5.

possibility of making a pivot-on-one, and the GLISM should then be less likely to get trapped at a local optimum.

Set partitioning problems with large, or huge, numbers of variables typically arise in a column generation context. This leads us to study the possibility of incorporating column generation in the integral simplex method, that is, a simplex method that maintains integrality by making pivots-on-ones only.

2 Column generation and implicit enumeration

Consider the set partitioning problem

$$
\begin{aligned}
[SPP_{\mathcal{N}}] \qquad z^* \;=\; & \min \sum_{j\in\mathcal{N}} c_j x_j \\
& \text{s.t.} \quad \sum_{j\in\mathcal{N}} a_j x_j = e, && \text{(3a)} \\
& \qquad x_j \in \{0,1\}, \quad j \in \mathcal{N}, && \text{(3b)}
\end{aligned}
$$

where $\mathcal{N}$ is a finite set of indices for the variables, and all other notations are defined as in SPP, with $\mathcal{N}$ replacing N. Assume that the problem is feasible, that the matrix $(a_j)_{j\in\mathcal{N}}$ has no zero rows or columns, and full rank. Let $SPP^{LP}_{\mathcal{N}}$ be the linear programming relaxation. The notations B, I, J, u, and Q will be used as before.

Further, we introduce the set $\mathcal{P} = \{(c_j, a_j) : j \in \mathcal{N}\} \subseteq Z \times \{0,1\}^m$. In a column generation situation, $|\mathcal{N}|$ is typically huge and the set $\mathcal{P}$ is typically described by constraints.

In an application there might be columns that are identical except for their costs, and in such a case, only the cheapest one will be contained in $\mathcal{P}$; hence $a_j \neq a_k$ whenever $j \neq k$. Assume that the columns defined by the set $N \subset \mathcal{N}$ are at hand, and let SPP_N^{LP} denote the linear programming relaxation of the corresponding restriction of $SPP_{\mathcal{N}}$.

In the following, we outline how the problem $SPP_{\mathcal{N}}$ can be solved by incorporating column generation in the integral simplex method. As described above, pivots-on-one are either non-degenerate or degenerate. Therefore, we distinguish between column generation that leads to non-degenerate and degenerate pivots-on-one.

To guarantee the finding of a global optimum, we consider an implicit enumeration procedure, similar to Thompson's GLISM. Our branching strategy is however new, and tailored to the column generation environment in the respect that the branchings also lead to restrictions on the entries of the columns not already at hand. (Alternative branching strategies may of course be possible.)

2.1 Column generation in the integral simplex method

At the root of the search tree, an initial set (possibly empty) of columns from the set $\mathcal{P}$ is at hand. By adding m artificial variables with large costs to the problem, we obtain an initial problem, $RSPP_0$. The artificial variables are indexed by the set S.

At subproblem k of the search tree, we consider the problem SPP_{N_k}, which contains the columns corresponding to the set $N_k \subseteq \mathcal{N} \cup S$. The branchings leading to this subproblem (see Section 2.2) give rise to fixations of variables, and because of these fixations the problem can be reduced further, both with respect to variables and constraints. The overall result of these reductions is represented by the sets $N_k^0 \subseteq \mathcal{N}$, $N_k^1 \subseteq \mathcal{N}$, and $M_k \subseteq M$, where the two former define the variables that are fixed to zero and one, respectively, and the latter denotes the set of constraints that remains in the problem.

The reduced problem is given by

$$
\begin{aligned}
[RSPP_k] \qquad z_k^* \;=\; & \min \sum_{j \in N_k^f} c_j x_j + \sum_{j \in N_k^1} c_j \\
& \text{s.t.} \quad \sum_{j \in N_k^f} a_{ij} x_j = 1, \quad i \in M_k, && \text{(4a)} \\
& \qquad\quad x_j \in \{0,1\}, \quad j \in N_k^f, && \text{(4b)}
\end{aligned}
$$

where $N_k^f = N_k \setminus \{N_k^0 \cup N_k^1\}$. Its linear programming relaxation is denoted by $RSSP_k^{LP}$.

The branchings in the tree also impose constraints on the columns to be generated. The branching technique suggested in Section 2.2 leads to restrictions that prevent any regeneration of columns and zero-fixations of entries in new columns. We denote by $A_k^0 \subseteq M$ the set of column entries that shall be fixed to zero when new columns are generated. Note that $A_k^0 \supseteq \{i \in M \mid \sum_{j \in N_k^1} a_{ij} = 1\}$ must hold, since one-fixations of variables eliminate constraints from the problem.

In the integral simplex method with column generation (ISMCG), for solving $RSSP_k$, the following steps are performed.

1. Initialize with a basis consisting of the $|M_k|$ artificial columns.

2. Perform as many non-degenerate pivots-on-one as possible.

3. Perform a non-degenerate column generation, that is, a column generation that enables a non-degenerate pivot-on-one (see Section 2.1.1). If successful, perform the pivot-on-one and go to Step 2.

4. If possible, perform a degenerate pivot-on-one and go to Step 2.

5. Perform a degenerate column generation, that is, a column generation that enables a degenerate pivot-on-one (see Section 2.1.2). If successful, perform the pivot-on-one and go to Step 2.

6. The current integral basic feasible solution is a local optimum. Let x_k^* denote the feasible solution to SPP_{N_k} obtained by augmenting the local optimal solution with the fixations given by N_k^0 and N_k^1. Terminate.

Here we prioritize non-degenerate pivots over degenerate, even at the expense of a column generation. This choice is of course optional, and may depend on the application considered.

It should be noted that all intermediate solutions encountered in ISMCG are integral and, if the artificial variables are all zero, feasible in the original problem $SPP_{\mathcal{N}}$. Hence, the method is primal all-integer, and, moreover, the sequence of objective values is non-increasing.

In the following, $(\bar{u}_i)_{i \in M_k}$ denotes the current basic dual solution to $RSPP_k^{LP}$, when a column generation is to be performed according to Step 2 or 4 described above.

2.1.1 Non-degenerate column generation

In order to generate a column that enables a non-degenerate pivot-on-one, we rely on the following result, which is shown by Balas and Padberg [1].

Theorem 3 *Let $\bar{x}$ be an integer basic feasible solution to $SPP_{\mathcal{N}}^{LP}$, associated with the basis B. Then there exists a basis that is adjacent to B and associated with an integer basic feasible solution different from $\bar{x}$, if and only if there exists $j \in \mathcal{N} \setminus I$ such that*

$$\bar{a}_{ij} = \begin{cases} 0 \text{ or } +1, \text{ for } i \in M \text{ such that } \bar{e}_i = 1, \\ 0 \text{ or } -1, \text{ for } i \in M \text{ such that } \bar{e}_i = 0, \\ +1, \text{ for at least one } i \in M \text{ such that } \bar{e}_i = 1. \end{cases} \tag{5}$$

The following corollary is an adaptation to our setting.

Corollary 4 *Given an integer basic feasible solution to $RSPP_k^{LP}$, associated with the basis B_k, obtained by making only pivots-on-one from a purely artificial initial basis, then a column (c_j, a_j), $j \in \mathcal{N}$, with $a_{ij} = 0$, $i \in A_k^0$, enables a non-degenerate pivot-on-one if and only if*

$$\bar{a}_{ij} \in \begin{cases} [0,1], \text{ for } i \in M \text{ such that } \bar{e}_i = 1, \\ [-1,0], \text{ for } i \in M \text{ such that } \bar{e}_i = 0 \end{cases} \tag{6}$$

holds.

Proof: For the subproblem $RSPP_k^{LP}$, the discrete choices in the first two cases in (5) can be replaced by the intervals given in (6), since B_k^{-1} is integral, because of the use of pivots-on-one only, $a_{ij} \in \{0, 1\}$, $i \in M_k$, and $(\bar{a}_{ij})_{i \in M_k} = B_k^{-1}(a_{ij})_{i \in M_k}$.

If the first two expressions in (5) hold, but the third does not, then the properties of the column would imply that the problem had an unbounded optimum, which is impossible since $SPP_{\mathcal{N}}^{LP}$ has a bounded optimum and $RSPP_k^{LP}$ is a restriction thereof.

Hence, the expressions (6) and (5) are equivalent under the given conditions. ■

The non-degenerate column generation problem can be stated as follows.

$$
\begin{aligned}
[NDCG_k] \qquad \bar{c}_p \;=\; & \min\; c - \sum_{i \in M_k} \bar{u}_i a_i \\
& \text{s.t.} \quad (c, a) \in \mathcal{P}, && \text{(7a)} \\
& \qquad\; a_i = 0,\; i \in A_k^0, && \text{(7b)} \\
& \qquad\; (c, a) \neq (c_j, a_j),\; j \in N_k^0, && \text{(7c)} \\
& \qquad\; \left(B^{-1}a\right)_i \in \begin{cases} [0, 1]\,, & \text{for } i \in M \text{ such that } \bar{e}_i = 1, \\ [-1, 0]\,, & \text{for } i \in M \text{ such that } \bar{e}_i = 0. \end{cases} && \text{(7d)}
\end{aligned}
$$

By assumption the columns are uniquely determined by their constraint coefficients, and therefore (7c) can be stated as the linear constraints

$$
\sum_{i \in M} a_{ij}\,(1 - a_i) + \sum_{i \in M} (1 - a_{ij})\,a_i \geq 1, \quad j \in N_k^0. \tag{8}
$$

If successful, the problem $NDCG_k$ yields a feasible column (c_p, a_p), with $\bar{c}_p < 0$, that enables a non-degenerate pivot-on-one. Note that none of the columns already available in $RSPP_k$ has these two properties; hence, such a generated column is not already at hand. The set N_k is augmented and the problem $RSPP_k$ is reoptimized by making a pivot-on-one on an entry in the new column.

The column generated can of course fulfil the restrictions (7b) and (7c) in other subproblems than the current, and it can in such a case be added to the set N_k in any such subproblem.

2.1.2 Degenerate column generation

According to Definition 2, a column enables a degenerate pivot-on-one if at least one of the updated entries in the degenerate rows equals one. The other restrictions on the column to be generated are as in the non-degenerate case. The degenerate column generation problem thus becomes as follows.

$$
\begin{aligned}
[DCG_k] \qquad \bar{c}_p \;=\; & \min\; c - \sum_{i \in M_k} \bar{u}_i a_i \\
& \text{s.t.} \quad (7a), (7b), (7c), \\
& \qquad \left(B^{-1}a\right)_i = 1 \text{ for some } i \in M \text{ such that } \bar{e}_i = 0. \qquad (9a)
\end{aligned}
$$

The restriction (9a) can be stated as the following linear constraints, with auxiliary binary variables $y_i,\ i \in M$ such that $\bar{e}_i = 0$, and C representing a sufficiently large number.

$$
\begin{aligned}
1 - C(1 - y_i) \le \left(B^{-1}a\right)_i \;&\le\; 1 + C(1 - y_i),\ i \in M \text{ such that } \bar{e}_i = 0 \qquad (10) \\
\sum_{i \in M:\ \bar{e}_i = 0} y_i \;&\ge\; 1 \qquad (11)
\end{aligned}
$$

If successful, the problem DCG_k yields a feasible column (c_p, a_p), with $\bar{c}_p < 0$, that allows a degenerate pivot-on-one. As in the non-degenerate case, such a column is new, and can be used in any other subproblem where it is feasible.

2.2 Implicit enumeration

The implicit enumeration procedure creates a search tree of subproblems using the branching technique to be described. (Proper alternative branching techniques can also be used.) Each branch in the search tree is defined by constraints (e.g. variable fixations), which, because of the special structure of the set partitioning problem, typically leads to a reduction of the problem, see Section 2.3.

As introduced in Section 2.1, a reduced problem is denoted by $RSPP_k$ and associated with four index sets. Fixations of variables are represented by N_k^0 and N_k^1, while zero-

fixations of coefficients, in the columns to be generated, are given by A_k^0. The set of rows that remain in the reduced problem is denoted by M_k.

Let L be the set of reduced subproblems of the form $RSPP_k$, not yet solved. Initially L consists of $RSPP_0$ only, for which $N_0^0 = N_0^1 = A_0^0 = \emptyset$ and $M_0 = M$. A reduced subproblem is solved by the method described in Section 2.1. To determine when it is impossible to find an improved solution to the current subproblem, a standard column generation problem, that is, $NDCG_k$ without constraints (7d), is used. This problem is referred to as CG_k.

The implicit enumeration procedure is given by the following steps.

1. If there are no more reduced subproblems $RSPP_k$ to solve, that is, if L is empty, then terminate. The best solution saved is an optimal solution to $SPP_{\mathcal{N}}$. Otherwise choose a problem from L.

2. Apply ISMCG to the subproblem and obtain a local optimal solution, x_k^*, with objective value z_k^*. Let Q_k be the set of variables that take the value one in x_k^*.

3. If all artificial variables are zero, that is, if $Q_k \cap S$ is empty, and the objective value z_k^* is the best found this far, then save the solution x_k^*.

4. If only artificial variables take the value one in the subproblem, that is, if $|Q_k \cap S| = |M_k|$, then cut the current branch and go to Step 1. In this case, N_k is empty and it is not possible to generate any new columns using $NDCG_k$ or DCG_k, and therefore there is no feasible solution to this subproblem.

5. If there are no negative reduced costs among the variables at hand in the subproblem, use the column generations problem CG_k to investigate if there are any negative reduced costs among the remaining variables, indexed by $\mathcal{N} \setminus N_k$. If there are none, then cut the current branch and go to Step 1, because there is no better feasible solution to this subproblem.

6. Perform a branching according to the strategy described in Section 2.3, for example. (In the branching, one needs to be careful since artificial variables may take the value one in the local optimal solution, see further Section 2.3). Let L_k' be the set of new subproblems, of the form SPP_{N_k}, that arise through the branching.

7. For each subproblem in L'_k, perform the following steps.

   (i) Reduce the subproblem SPP_{N_k} according to the steps described in Section 2.3, giving $RSPP_k$, which is represented by N_k^0, N_k^1, A_k^0, and M_k.

   (ii) If M_k is empty, cut the current branch, otherwise add $RSSP_k$ to L.

   Go to Step 1.

This procedure will finitely find a global optimum to $SPP_{\mathcal{N}}$, provided that the branching strategy is well chosen. An example of such a strategy is given below. It should be noted that the branching technique used in [10] is not appropriate in our setting with column generation, since it requires the explicit knowledge of all non-basic variables with negative reduced costs. Our branching strategy is however related to the one used by Thompson, with zero- and one-fixations of variables replacing each other.

2.3 A branching technique

In the implicit enumeration procedure, Step 7 should make use of a branching strategy that excludes the local optimum to the current reduced subproblem, $RSPP_k$, from further consideration. The strategy suggested here does that, and at the same time, it partitions the remaining feasible set. The branching is first described for the case when there is no artificial variable that equals one in the local optimum to the subproblem, and it is then modified to suit the general case. To avoid introducing much notation and to make the description easy, our branching strategy is mainly explained by applying it to the local optimum obtained in Example 1. We here assume that the given problem is an initial subproblem, $RSPP_0$, and that more columns can be generated.

Generally, we consider the variables that take the value one in a local optimum, and for each of these we create a branch where the variable is set to zero. We thus exclude the local optimum from all the branches. In order to partition the feasible set, we also make one-fixations of the variables that are set to zero in the branches which have already been created from this subproblem. At the local optimum in Example 1, the two variables x_1 and x_7 take the value one. We then create a first branch with $x_1 = 0$ and a second with $x_7 = 0$ and $x_1 = 1$, thus excluding the local optimum and partitioning the feasible set.

Each of the two branches is split into two sub-branches by using a constraint in $RSPP_k$ that contains the variable that is being fixed to zero. The chosen constraint can then become fulfilled either by one of the variables at hand, which is represented by the left branch, or by a variable to be generated, which is represented by the right branch. Hence, in a column to be generated, the entry in this constraint must be zero in the left sub-branch, and zero or one in the right. This sub-branching thus partitions the feasible set further. In the example, we use the fourth constraint in the first branch, and the fifth constraint in the second one. The branching is illustrated in Figure 2.

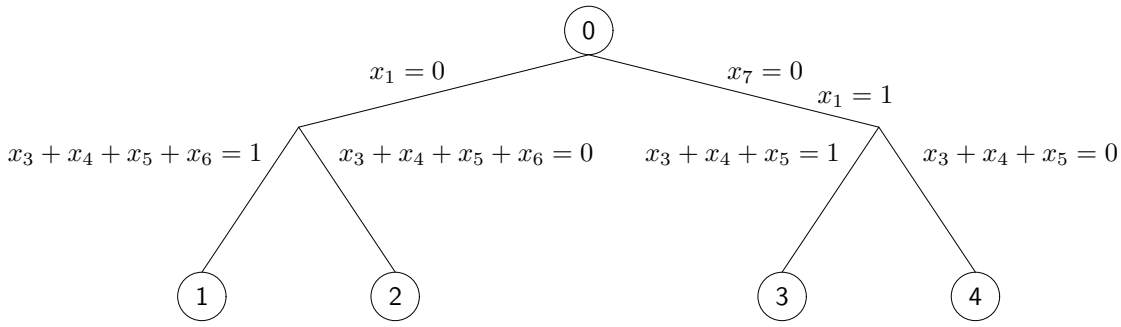

Figure 2: A branching strategy that can be used in Step 7 in the implicit enumeration procedure.

For each sub-branch in the search tree in Figure 2, we will now describe the reduction of the corresponding subproblem and form the sets N_k^0, N_k^1, A_k^0, and M_k, that represent the fixations arising from the branching constraints and the reduction.

- **Sub-branch 1:** The branching constraint $x_3 + x_4 + x_5 + x_6 = 1$ coincides with the fourth original constraint if we fix $x_1 = 0$ and only generate new columns with $a_4 = 0$. These fixations are represented by $N_1^0 = \{1\}$ and $A_1^0 = \{4\}$. No reduction of the problem is possible, N_1^1 remains empty, and $M_1 = M$.

- **Sub-branch 2:** To fulfil the branching constraints $x_1 = 0$ and $x_3 + x_4 + x_5 + x_6 = 0$, all the variables involved must be fixed to zero; this is represented by the set $N_2^0 = \{1, 3, 4, 5, 6\}$. No reduction of the problem is possible, N_2^1 and A_2^0 remain empty, and $M_2 = M$.

- **Sub-branch 3:** Here the fixation $x_1 = 1$ enables a reduction of the problem. All variables at hand that are somewhere contained in the same constraint as x_1 can be removed from the problem, and no new variable that would be contained in any of these constraints is allowed to be generated. Since also $x_7 = 0$, the zero-fixations of variables are represented by $N_3^0 = \{3, 4, 5, 6, 7, 8, 9, 10, 11\}$. Further, $N_3^1 = \{1\}$, and $M_3 = \{1, 5\}$. Finally, $A_3^0 = \{2, 3, 4, 5\}$, where $a_5 = 0$ must hold since the sub-branching is made over the fifth constraint. (Note that in this sub-branch there is no feasible solution.)

- **Sub-branch 4:** Here the fixation $x_1 = 1$ enables the same reduction of the problem as in the third sub-branch, and therefore $N_4^0 = \{3, 4, 5, 6, 7, 8, 9, 10, 11\}$. Further, $N_4^1 = \{1\}$, $M_4 = \{1, 5\}$, and $A_4^0 = \{2, 3, 4\}$.

Note that when the suggested branching is applied repeatedly, the sets N_k^0, N_k^1, and A_k^0 are expanded with new elements, while the set M_k is reduced.

We now consider the case where at least one artificial variable equals one in the local optimum to the subproblem. In this case it is crucial that the branching is performed so that the first branches correspond to zero-fixations of structural variables. This can always be accomplished by an appropriate sorting of the variables.

Suppose, for example, that the structural variables x_1 and x_2 and the artificial variables x_1^a and x_2^a take the value one in a local optimum. When using the branching strategy that was applied to the local optimum in Example 1, with the variables properly sorted, we obtain the four branches shown in Figure 3.

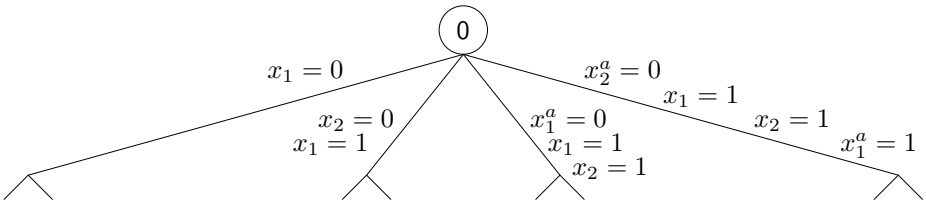

Figure 3: The branching strategy used in Step 7, before it is adjusted to handle the presence of artificial variables among the one-valued variables in a local optimum.

Clearly, any branch where an artificial variable is fixed to one is of no interest, and

need not be created. Furthermore, among the remaining branches there will be exactly one which has a zero-fixation of an artificial variable.

It is however not possible to make an explicit zero-fixation of an artificial variable, since the artificial variables form the initial basis in the ISMCG when the subproblem is solved. This fixation can instead be made implicitly by instead of making the zero-fixation, letting the artificial variable under consideration leave the basis in the first pivot in ISMCG and never let it enter the basis again. Since ISMCG starts from a purely artificial and non-degenerate basis, it is always possible to make a non-degenerate pivot-on-one with any arbitrary artificial variable leaving the basis in the first iteration. The artificial variable to be fixated will thus become zero, and it will remain zero since it is forbidden to enter the basis again.

In order to retain the zero-fixation of an artificial variable in the entire sub-tree that originates from a sub-branch with this artificial variable fixed to zero, the same small modification of ISMCG is used throughout the sub-tree. This is possible, since there can infact never be more than one zero-fixation of an artificial variable in any sub-branch. To explain this, we first observe that the constraint corresponding to a zero-fixated artificial variable must, in a local optimum, become fulfilled by a structural variable. Second, due to the special sorting of the one-valued variables, whenever a new artificial variable is to be fixed to zero, all one-valued structural variables in the local optimum are to be fixed to one. Hence, all previously zero-fixated artificial variables and their corresponding constraint must have been removed. We conclude that, anywhere in the search tree, at most one artificial variable is zero-fixated actively.

The branching strategy outlined in this section enables a branching to be performed whenever the local optimum found has not been identified as a global optimum to the current subproblem, or the subproblem has not been identified as being infeasible. The branching eliminates the local optimum to the subproblem from further consideration and partitions the remaining feasible set. These properties are sufficient to guarantee that a global optimum to $SPP_{\mathcal{N}}$ will be found finitely, provided that cycling is prevented in ISMCG.

3 Concluding remarks

We have outlined how to tailor the column generation principle to the integral simplex method for set partitioning problems. The special properties of the set partitioning polytope makes it possible to generate only columns that will lead to pivots between integer solutions. By combining our column generation principle with implicit enumeration, we obtain a novel primal all-integer method.

The properties of the method outlined differs from those of traditional branch-and-price, in which only columns needed in the process of solving the linear programming relaxation are generated at the root node of the search tree. These columns may, of course, be of little use for solving the integer problem. Further, down the search tree, all columns generated are those needed for solving the current linear programming relaxations, although they will become better suited for the integer problem, and some of them will eventually be those required to solve the integer problem.

Our main contribution is the adaptation of the column generation principle to the integral simplex method for set partitioning problems. A price that has to be paid for preserving integrality is that the column generation is more complex than in the traditional case, and therefore likely to be computationally more demanding. (Some complex column generation problems have been solved successfully by constraint programming techniques, see e.g. [6], which could therefore be a viable solution alternative for column generation in the integral simplex method.)

The implicit enumeration and branching strategies presented should be seen only as suggestions for dealing with the difficulty of local optima that are not global. Another possibility for handling this difficulty could be to allow degenerate pivots on minus-one-entries. However, there seems to be no known way for preventing cycling in the integral simplex method, and this could be even more cumbersome if pivots on minus-one-entries are allowed. Our choice of branching strategy is motivated by the fact that it leads to zero-fixations of entries in columns to be generated. In many applications of column generation, these fixations should be easy to take into account.

Meta-heuristics are useful in many applications of combinatorial optimization because of their ability to produce acceptable solutions in reasonable computing times. One may note that the integral simplex method (with column generation) can be regarded as a local

search method, which can be trapped at a local optimum, as given in Definition 3. An alternative to performing a tree search would thus be to use a meta-heuristic approach, for example, to employ a tabu strategy for escaping from local optima. When employing such a strategy it could also be advantageous to allow variables with positive reduced costs to enter the basis or to allow degenerate pivots on minus-one-entries. The tabu strategy can then preferably be designed with the additional purpose of preventing cycling.

Our intention with this paper was merely to present a new and original approach to column generation for set partitioning. There are certainly several opportunities for further research and development into column generation customized to maintain integrality, along the lines discussed here. One of the questions of interest is to study if the use of pivots-on-one is useful for solving problems similar to the set partitioning problem, for example if side constraints of special kind are added.

Acknowledgment

The authors were financially supported by a grant from the Swedish Research Council (grant no. 621-2005-2874). We also thank the referee, who made valuable comments.

References

[1] E. Balas and M. W. Padberg. On the set-covering problem. *Operations Research*, 20:1152–1161, 1972.

[2] E. Balas and M. W. Padberg. On the set-covering problem: II. An algorithm for set partitioning. *Operations Research*, 23:74–90, 1975.

[3] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, and P. H. Vance. Branch-and-price: column generation for solving huge integer programs. *Operations Research*, 46:316–329, 1998.

[4] J. Hellstrand, T. Larsson, and A. Migdalas. A characterization of the uncapacitated network desing polytope. *Operations Research Letters*, 12:159–163, 1992.

[5] K. L. Hoffman and M. Padberg. Solving airline crew scheduling problems by branch-and-cut. *Management Science*, 39:657–682, 1993.

[6] U. Junker, S. E. Karisch, N. Kohl, B. Vaaben, T. Fahle, and M. Sellmann. A framework for constraint programming based column generation. In *Proceedings of the 5th International Conference on the Principles and Practice of Constraint Programming (CP)*, volume 1713 of *Springer Lecture Notes in Computer Science*, pages 261–274, Berlin, 1999. Springer.

[7] M. E. Lübbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53:1007–1023, 2005.

[8] T. Sastry. One and two facility network design revisited. *Annals of Operations Research*, 108:19–31, 2001.

[9] M. Savelsbergh. A branch-and-price algorithm for the generalized assignment problem. *Operations Research*, 45:831–841, 1997.

[10] G. L. Thompson. An integral simplex algorithm for solving combinatorial optimization problems. *Computional Optimization and Application*, 22:351–367, 2002.

[11] V. A. Trubin. On a method of solution of integer linear programming problems of a special kind. Translated by V. Hall. *Soviet Math Doklady*, 10:1544–1546, 1969.

[12] W. E. Wilhelm. A technical review of column generation in integer programming. *Optimization and Engineering*, 2:159–200, 2001.

[13] V. A. Yemelichev, M. M. Kovalev, and M. K. Kravtsov. *Polytopes, graphs and optimisation.* Translated by G. H. Lawden. Cambridge University press, Cambridge, 1984.