All-integer column generation for set partitioning: Basic principles and extensions

Elina Rönnberg and Torbjörn Larsson

Original Publication:

Elina Rönnberg and Torbjörn Larsson, All-integer column generation for set partitioning: Basic principles and extensions, 2014, European Journal of Operational Research, (233), 3, 529-538.
http://dx.doi.org/10.1016/j.ejor.2013.08.036

All-integer column generation for set partitioning: basic principles and extensions

Elina Rönnberg* and Torbjörn Larsson*

August 2013

Abstract: Column generation, combined with an appropriate integer programming technique, has shown to be a powerful tool for solving huge integer programs arising in various applications. In these column generation approaches, the master problem is often of a set partitioning type.

The set partitioning polytope has the quasi-integrality property, which enables the use of simplex pivots for finding improved integer solutions, each of which is associated with a linear programming basis. By combining such pivots with column generation, one obtains a method where each found solution to a restricted master problem is feasible, integer, and associated with a dual solution that can be used in a column generation step.

This paper presents a framework for such an all-integer column generation approach to set partitioning problems. We give the basic principles of all-integer pivots and all-integer column generation. We also state optimality conditions and introduce means for preserving a basis in the event that a heuristic is applied to the master problem. These extensions introduce flexibility in the design of a specific solution scheme of this kind, and with proper settings optimal or approximate solutions can be sought for.

Keywords: Integer programming, Column generation, Set partitioning, Quasi-integrality, All-integer pivots, Metaheuristics

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

1 Introduction

Column generation is a linear programming method that is designed for solving problems that have huge numbers of variables, but also a certain structure. When complemented with a suitable integer programming technique, column generation has proven to be successful in solving many large scale integer programs. A well known strategy of this type is branch-and-price, where column generation is embedded in a branch-and-bound scheme, see Barnhart et al. [3] and Vanderbeck [21]. For surveys of column generation and applications within integer programming, see Lübbecke and Desrosiers [10] and Wilhelm [22]. Our work originates from the fact that many applications of column generation give rise to master problems of a set partitioning type, as for example aircrew rostering problems, see Gamache et al. [7], and the generalised assignment problem, see Barnhart et al. [3].

The feasible set of the linear programming relaxation of the set partitioning problem possesses the *quasi-integrality property*, which means that for the polytope of feasible solutions, every edge of the convex hull of the integer points is also an edge of the polytope itself. This property makes it possible to utilise linear programming techniques for finding improved integer solutions, an idea which was first explored by Trubin [20] in the sixties. It was later somewhat more thoroughly described and named *the integral simplex method* in Yemelichev et al. [23]. The integral simplex method only acts as a local search method, but complemented with a branching strategy, as in the implementation made by Thompson [19], the finding of an optimal solution can be ensured. In Rönnberg and Larsson [16], based on results in Balas and Padberg [1], it is shown how the column generation principle can be integrated with the integral simplex method, and how a branching strategy can be used to ensure the finding of an optimal solution.

The work to be presented in this paper is a continuation of the research published in [16], [15] and [14], although we here consider more general pivots than those used in the integral simplex method and [16]; these are called *all-integer pivots*. We develop the basic principles of *all-integer column generation* for set partitioning problems, together with several extensions.

To be more precise, we consider column generation problems that are consistent with all-integer pivots, state optimality conditions for the integer problem and use the concept of *over-generation of columns*, which means generating not only an optimal column but several, by generating near-optimal solutions to a regular column generation subproblem. This can be used as an additional way to augment the master problem, or for generating enough columns for guaranteeing that the master problem can provide an integer optimal solution.

Further, so called *surrogate columns* are introduced. These facilitate the use of combinatorial heuristics on the master problem within a framework of all-integer column generation. This feature is of interest because strategies for how to combine column generation and heuristics are not very well explored in the literature, although the development of such strategies seem to attract more attention; see for example Joncour et al. [8], which gives an overview, and also the SearchCol project [11] on this topic.

Although the overall approach is primarily introduced as being of a meta-heuristic nature, criteria for determining whether a solution is optimal or near-optimal are also available. Key characteristics of the approach are that all successively found solutions are feasible and integral, and that the integrality requirement on the variables are handled differently from what has previously been described in the column generation literature.

The outline of the paper is as follows. Section 2 introduces notations and other prerequisites. Section 3 presents the foundations and principles of all-integer column generation for set partitioning: the quasi-integrality property, all-integer pivots, and all-integer column generation. The pivoting step naturally separates into the non-degenerate and degenerate cases, which aim at a primal and a dual improvement of the solution respectively. The same cases hold for the column generation step, and we show how the all-integer column generation subproblem can be tailored to produce columns enabling either non-degenerate or degenerate pivots.

Section 4.1 derives the surrogate column pivot, which enables making an all-integer pivot from an integer feasible solution with a known basis, to another integer feasible solution for which no basis is known. Section 4.2 gives optimality conditions for the integer problem and outlines the over-generation strategy. In Section 4.3, the generalised assignment problem is used to illustrate the formulation of the all-integer column generation subproblems and how to transform the subproblems into a more suitable form. This transformation is believed to be of critical importance from a computational point of view. Also presented in this section are some simple computational experiments that illustrate the behaviour of the core components of the all-integer column generation approach. Some reflections on the overall methodology follows in Section 5, together with some ideas for future work.

Before proceeding with our presentation we would like to mention that one of the reviewers drew our attention to an interesting parallel work of Zaghrouti et al. [24]. The similarity to our work is that they utilise the quasi-integrality property to come up with an efficient way to generate linear programming pivots leading to integer extreme points. For this purpose they use a direction-finding subproblem that can be solved efficiently and they present excellent computational results for set partitioning problems with up

to 500000 variables. The main difference is that [24] does not consider a column generation setting.

2 Problem formulation and notation

The set partitioning problem to be solved is

$$
\begin{aligned}
[\mathrm{SPP}_{\mathcal{N}}] \qquad z^* \;=\; & \min \quad \sum_{j\in\mathcal{N}} c_j x_j \\
& \text{s.t.} \quad \sum_{j\in\mathcal{N}} a_{ij} x_j = e_i, \quad i \in M \\
& \qquad\quad x_j \in \{0,1\}, \quad j \in \mathcal{N}.
\end{aligned}
$$

Here, $\mathcal{N}$ and $M = \{1,\ldots,m\}$ are the sets of indices for the variables respective constraints. Further, $c_j \in \mathbf{R}$, $j \in \mathcal{N}$, $a_{ij} \in \{0,1\}$, $i \in M$, $j \in \mathcal{N}$, and $e_i = 1$, $i \in M$. We assume that $|\mathcal{N}| \geq m$ and that the problem is feasible. Let $a_j = (a_{1j},\ldots,a_{mj})^{\mathrm{T}}$, $j \in \mathcal{N}$, and assume, with no loss of generality, that the matrix $(a_1,\ldots,a_{|\mathcal{N}|})$ has no zero column and full rank.

In a column generation setting, problem $\mathrm{SPP}_{\mathcal{N}}$ is referred to as the complete master problem. Let $\mathcal{P} = \{(c_j, a_j) : j \in \mathcal{N}\} \subseteq Z \times \{0,1\}^m$ be the set of columns. This set typically corresponds to the set of feasible solutions to an optimization problem and it is typically huge. In an application there might be columns that are identical except for their costs, and in such cases, only the cheapest one will be contained in $\mathcal{P}$; hence $a_j \neq a_k$ whenever $j \neq k$.

Let SPP_N be a restricted set partitioning problem, or restricted master problem, where the variables with indices $N = \{1,\ldots,n\} \subset \mathcal{N}$ are at hand. It is assumed that $n \geq m$, that SPP_N is feasible, and that the matrix $A = (a_1,\ldots,a_n)$ has full rank. An optimal solution and the optimal objective value of SPP_N are denoted by $\tilde{x}$ and $\tilde{z}$, respectively.

Let SPP_N^{LP} be the linear programming relaxation of SPP_N, with $x_j \in \{0,1\}$ replaced by $x_j \geq 0$, $j \in N$. Because the feasible set is contained within the unit hypercube, all feasible integer points will be extreme points of the polytope describing the feasible set of SPP_N^{LP}. For an extreme point to SPP_N^{LP}, denote a basis by $B \subseteq \{a_1,\ldots,a_n\}$; if an extreme point is degenerate, it is associated with more than one basis. For a given basis, let I and J denote the corresponding sets of basic and non-basic columns respectively, let $x_B = (x_j)_{j\in I}$ and $x_N = (x_j)_{j\in J}$ be the corresponding basic and the non-basic components of x, and let $Q \subseteq I$ be the set of basic variables that take the value one. The notations B, I, J and Q will sometimes be used together with a superscript referring to the extreme point with which they are associated.

Further, let $\bar{e} = B^{-1}e$, where $e = (e_1, \ldots, e_m)^{\mathrm{T}}$, and $\bar{a}_j = B^{-1}a_j$, $j \in N$, be the updated right-hand-side and constraint columns respectively. The complementary dual solution is $u^{\mathrm{T}} = c_B^{\mathrm{T}} B^{-1}$, where $c_B = (c_j)_{j \in I}$, and the reduced costs are $\bar{c}_j = c_j - u^{\mathrm{T}} a_j$, $j \in J$.

For SPP_N^{LP}, with basis B and dual solution u, the column generation subproblem

$$\begin{array}{lrll} \text{[CG]} & \bar{c}_p = & \min & c - \sum_{i \in M} u_i a_i \\ & & \text{s.t.} & (c, a) \in \mathcal{P}, \end{array}$$

is used to produce a column (c_p, a_p) that can be added to the restricted master problem. For future reference, we note that if the column generation is made using a dual feasible solution with respect to the problem SPP_N^{LP}, then the generated column (c_p, a_p) is never already included in the restricted master problem SPP_N (unless dual feasibility holds also in $\mathrm{SPP}_{\mathcal{N}}^{LP}$, in which case $\bar{c}_p = 0$). If however, the column generation is made using a dual infeasible solution with respect to SPP_N^{LP}, then it might happen that the column (c_p, a_p) is already included in the restricted master problem.

3 Background

This section presents the theoretical background needed for the results to be presented in this paper. The first results are on the polyhedral structure of the set partitioning problem and trace all the way back to the sixties [20], followed by recent results from [16] that introduce a way of utilising this polyhedral structure in a column generation setting.

3.1 Quasi-integrality and all-integer pivots

One way of finding an improved solution to the restricted master problem is to use the integrality preserving pivots to be presented in this section. This approach is possible because the polytope of feasible solutions of SPP_N^{LP} has the quasi-integrality property, which implies that all integer extreme points can be reached by making simplex pivots between integer extreme points; this is referred to as the *integral simplex method*. The difficulty lies in the finding of bases that enable such pivots that lead to improved solutions, and if no such basis is found, the method will stop at a local optimum.

The following definition is from Yemelichev et al. [23].

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.*

The quasi-integrality property holds if each pair of integer extreme points belongs to an integral face of the polytope describing the set of feasible solutions to the linear programming relaxation of the integer problem [23]. By an integral face is meant a face that has the integrality property, which holds if the face can be described by constraints of which the matrix is totally unimodular. The following theorem was first shown by Trubin [20].

Theorem 1 *The feasible polytope of SPP_N^{LP} is quasi-integral.*

A movement between two integer extreme points can be achieved algebraically through a non-degenerate simplex pivot on a one entry. Not all bases associated with an integer extreme point enable such a pivot, and because of degeneracy it can be difficult to find a basis that does. In the implementation of the integral simplex method made by Thompson [19], which is the only one that we know of, the method starts from an artificial basis and then performs simplex pivots on one-entries. A pivot on a one-entry is henceforth referred to as a *pivot-on-one*, and it can be either degenerate or non-degenerate. When making pivots-on-one only, it is possible to reach a basis that is non-optimal and where it is not possible to perform a pivot-on-one; this situation is referred to as a local optimum. Another difficulty is that there is no known way to prevent cycling in the integral simplex method. For more information on quasi-integrality and the integral simplex method, see [20],[23], [19], and [16].

We consider a slight extension of pivots-on-one, called *all-integer pivots*, see Rönnberg [15]. It is motivated by the results below. For future reference, we begin with the following definition.

Definition 2 *A linear programming basis B is* unimodular *if $|det(B)| = 1$.*

Unimodular bases are of special interest when working with integer solutions to set partitioning problems, as stated in the following lemma, the proof of which is elementary using Cramer's rule.

Lemma 1 *If some basis for an extreme point to SPP_N^{LP} is unimodular, then the extreme point is integral.*

This sufficient condition for an extreme point to be integral is not necessary however, and there are even cases where none integer extreme point is associated with a unimodular basis; for an example of this, see [15].

To preserve integrality when pivoting, a sufficient condition is to restrict the pivots to be between unimodular bases only, and this observation naturally leads to the following definition.

Definition 3 *Given a unimodular basis associated with a feasible solution to* SPP_N^{LP} *and a* $s \in J$ *such that* $\bar{c}_s < 0$. *An* all-integer pivot *is,*

(i) in the non-degenerate *case a pivot made on an entry* $\bar{a}_{rs} = 1$ *such that*

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

(ii) in the degenerate *case a pivot made on an entry* $\bar{a}_{rs} = \pm 1$ *such that* $\bar{e}_r = 0$.

The following theorem states a sufficient condition for a sequence of pivots to be between integer solutions only. The proof is straightforward, using that the determinant of the pivot matrix (see e.g. Murty [12, page 141]) is ± 1, and can be found in [15].

Theorem 2 *Starting from a feasible unimodular basis for* SPP_N^{LP}*, a sufficient condition for a sequence of pivots to yield a path of integer extreme points is that the pivots are all-integer.*

A non-degenerate pivot-on-one from a basis which is not unimodular can lead to a non-integer extreme point, as shown in an example in [15].

If the column chosen to enter the basis has a negative reduced cost, then a non-degenerate pivot-on-one at a unimodular basis associated with an integer extreme point will lead to another integer extreme point and yield a primal improvement. A degenerate pivot, on the other hand, will not yield any primal improvement; instead the purpose of a degenerate pivot, if it is not also dual degenerate, is to decrease the dual infeasibility.

In the integral simplex method implemented by Thompson [19], only pivots on positive entries are allowed to be made. Our extension, to also allow the degenerate pivots to be on a minus one entry, increases the number of possible pivots and could therefore decrease the risk of getting trapped at a local optimum. We do not know of an anti-cycling rule for the all-integer pivots, and we consider this to be a challenging topic for future research.

3.2 All-integer column generation

We here present the *all-integer column generation* strategy for augmenting the restricted master problem with new columns. This strategy employs a column generation problem that is restricted to find only such columns that will enable all-integer pivots, either non-degenerate or degenerate. All-integer column generation relies on that a basis, and thereby also a dual solution, for the current integer extreme point is available. The results below are based on those in Sections 2.1.1 and 2.1.2 of Rönnberg and Larsson [16].

To begin with the non-degenerate case, the corollary to follow was originally presented as Corollary 4 in [16], which in turn is an adaptation of Corollary 2.1 of [1] to the setting used in [16], and now also in this paper. The corollary provides necessary and sufficient conditions for a column to enable a non-degenerate pivot-on-one.

Corollary 1 *Given an integer extreme point to SPP_N^{LP} and an associated unimodular basis B, then, a column (c_s, a_s), $s \in \mathcal{N} \setminus I$, enables a non-degenerate pivot-on-one if and only if*

$$\bar{a}_{is} \in [-1 + \bar{e}_i, \bar{e}_i] = \begin{cases} [0,1], & i \in M: \ \bar{e}_i = 1, \\ [-1,0], & i \in M: \ \bar{e}_i = 0 \end{cases}$$

holds.

Due to the unimodularity of B and the integrality of a_s, the value of $\bar{a}_s$ is always integral. By adding the condition of Corollary 1 as side constraints to CG, the non-degenerate all-integer column generation problem, AICG-ND, can be stated as follows.

$$\begin{aligned} \text{[AICG-ND]} \qquad \bar{c}_p \ = \quad & \min \quad c - \sum_{i \in M} u_i a_i \\ & \text{s.t.} \quad (c,a) \in \mathcal{P}, \\ & \quad \left(B^{-1}a\right)_i \in \begin{cases} [0,1], & i \in M: \ \bar{e}_i = 1, \\ [-1,0], & i \in M: \ \bar{e}_i = 0 \end{cases} \end{aligned}$$

If $\bar{c}_p < 0$ holds for a solution to (c_p, a_p) to AICG-ND, then this column can be added to SPP_N and there enable a non-degenerate pivot-on-one, by letting its corresponding variable enter the basis.

In order to enable a degenerate all-integer pivot with a variable x_s entering the basis, it is sufficient to find a pivot element $\bar{a}_{rs} = \pm 1$ for a row $r \in M$

such that $\bar{e}_r = 0$. Adding this requirement to CG, the degenerate all-integer column generation problem, AICG-D, becomes as follows.

$$\begin{array}{lrll} \text{[AICG-D]} & \bar{c}_p = & \min & c - \sum_{i \in M} u_i a_i \\ & & \text{s.t.} & (c, a) \in \mathcal{P}, \\ & & & \left(B^{-1}a\right)_i = \pm 1 \text{ for some } i \in M: \ \bar{e}_i = 0 \end{array}$$

If $\bar{c}_p < 0$ holds for a solution (c_p, a_p) to AICG-D, then this column can be added to SPP_N and there enable a degenerate pivot-on-one, by letting its corresponding variable enter the basis. In [16] it is shown how the extra restriction added in AICG-D can be stated as linear constraints by using auxiliary binary variables.

For a specific application of all-integer column generation to the problem $\text{SPP}_{\mathcal{N}}$, where the column set $\mathcal{P}$ is described by regular constraints, the side constrained column generation problems AICG-ND and AICG-D can be restated as linear integer programs, in a straightforward way.

4 Extending the all-integer column generation framework

The results presented above provide a platform for an all-integer column generation approach for set paritioning problems. The possibilites for how to design an algorithm based only on this approach are however limited. This section presents the theoretical foundations for extensions which give more options in the design of an all-integer column generation algorithm. The first extention presented gives means for how to combine this approach with any method for finding improved integer feasible solutions to the restricted master problem. The second extension provides optimality conditions for when the restricted master problem contains all the columns necessary for finding an optimal solution to $\text{SPP}_{\mathcal{N}}$. These optimality conditions are used to derive a column over-generation strategy for guaranteeing that an optimal solution can be found without the use of a branching strategy.

As observed in a computational experiment described in Section 4.3.3, the linear integer programming formulations of AICG-ND and AICG-D can be poorly conditioned due to ill-conditioning of the basis matrix B. These complications can however be overcome by performing a transformation introduced and illustrated on the generalised assignment problem in Section 4.3.2.

4.1 Surrogate column pivots

The subproblem used in column generation relies on dual information from the restricted master problem, and if a basis for the current extreme point is at hand, this dual information is readily available. In this section, it is shown that if an improved integer solution to SPP_N is at hand, without an associated basis for this extreme point being known, then a basis can be constructed by performing a non-degenerate pivot-on-one.

We assume that a current integer extreme point x^c and an associated unimodular basis B^c are known. Further, let x^n be another integer extreme point, to be visited next but for which there is no known basis. The latter point could typically have been found by applying a heuristic to SPP_N. The superscripts c and n will in the following be used to refer to various quantities related to the integer extreme points x^c and x^n respectively.

The following lemma describes the requirements for a non-degenerate pivot-on-one to lead to a unimodular basis at x^n.

Lemma 2 *Let x^c and x^n be two integer extreme points to SPP_N^{LP}. A pivot on the entry $\bar{a}^c_{rs}$ from the unimodular basis B^c at x^c is a non-degenerate pivot-on-one leading to a unimodular basis at x^n if and only if*

$$\bar{a}^c_{is} = \begin{cases} 1, & i = r \\ \bar{e}^c_i - \bar{e}^n_i, & i \in M \setminus \{r\}. \end{cases}$$

Proof: For a non-degenerate pivot-on-one $\bar{a}^c_{rs} = \bar{e}^c_r = 1$ holds, and by definition of a simplex pivot, for $i \in M \setminus \{r\}$, $\bar{e}^n_i = \bar{e}^c_i - (\bar{a}^c_{is}/\bar{a}^c_{rs})\,\bar{e}^c_r = \bar{e}^c_i - \bar{a}^c_{is}$, that is, $\bar{a}^c_{is} = \bar{e}^c_i - \bar{e}^n_i$. ■

If there is no such updated column, then it is not possible to pivot directly from the basis B^c of x^c to x^n. By augmenting SPP_N with a properly chosen auxiliary variable, y, whose column fulfils the requirements of Lemma 2, it is however possible to make a pivot leading to a unimodular basis B^n at an extreme point $(x, y)^n$ of the augmented problem. As shown below, the auxiliary column will always be of the kind specified in the following definition.

Definition 4 *Let $S \subseteq N$ with $|S| \geq 2$. Then a positive linear combination of the columns (c_j, a_j), $j \in S$, is called a* surrogate column.

The name surrogate column stems from that a linear combination of constraints is known as a surrogate constraint, see e.g. Taha [18], and that a linear combination of columns is the dual correspondence thereof. A simplex

pivot on a surrogate column corresponds to moving through the relative interior of the feasible set, and in linear programming, this idea is far from new; already in 1951 the so-called feasible direction method was introduced by Brown and Koopmans [5]. Further work along this line has been carried out for example by Eiselt and Sandblom [6] and Murty and Fathi [13].

If the surrogate column shall be of a set partitioning type, the only viable alternative is that it is a sum of orthogonal original columns. The value of a surrogate variable can easily be re-expressed in terms of values of the original variables, as shown in the following lemma.

Lemma 3 *Let the columns* $\{a_j\}_{j \in S \subseteq N}$ *be orthogonal and suppose that* SPP_N^{LP} *is augmented with the surrogate column*

$$(c^y, a^y) = \sum_{j \in S} (c_j, a_j)$$

and its corresponding variable, denoted by y*. Let* $(\hat{x}, \hat{y}) \in \mathbf{R}^{n+1}$ *be an integer solution to the augmented problem. Then* $x = P\left((\hat{x}, \hat{y}), S\right) \in \mathbf{R}^n$*, with*

$$x_j = P_j\left((\hat{x}, \hat{y}), S\right) = \begin{cases} \hat{y}, & j \in S \\ \hat{x}_j, & j \in N \setminus S \end{cases}$$

is an integer solution to SPP_N^{LP}*.*

The theorem to follow is instrumental for the construction of a surrogate column that represents the movement from x^c to x^n.

Theorem 3 *For* SPP_N^{LP}*, assume that an integer extreme point* x^c*, with an associated unimodular basis* B^c*, is known. Further, let* x^n *be another integer solution. Suppose that* SPP_N^{LP} *is augmented with the surrogate column*

$$(c^y, a^y) = \sum_{j \in J^c \cap Q^n} (c_j, a_j)$$

and its corresponding variable, denoted by y*. Then, a pivot with the column* a^y *entering the basis is a non-degenerate pivot-on-one leading to a unimodular basis* B^n *at an integer extreme point* $(\hat{x}, \hat{y})^n \neq (x^c, 0)$ *of the augmented problem. Furthermore,* $x^n = P\left((\hat{x}, \hat{y})^n, J^c \cap Q^n\right)$*.*

Proof: In a feasible integer solution, the columns of the variables that take the value one will cover each row exactly once, and when an all-integer pivot

is performed, the variables that change value from zero to one, and from one to zero, must cover the same rows, that is,

$$\sum_{j\in(I^c\setminus Q^c)\cap Q^n} a_j + \sum_{j\in J^c\cap Q^n} a_j = \sum_{j\in Q^c\cap(I^n\setminus Q^n)} a_j + \sum_{j\in Q^c\cap J^n} a_j.$$

Lemma 2 states the necessary properties for an updated column, if pivoted into the basis, to yield a non-degenerate pivot-on-one from x^c to x^n. Using the notation a_{j_i} to refer to the column of the ith basic variable, the conditions of Lemma 2 can be rewritten as

$$\bar{a}^c_{is} = \begin{cases} 1, & j_i \in Q^c \cap J^n, \\ \bar{e}^c_i - \bar{e}^n_i = 1 - 0 = 1, & j_i \in Q^c \cap (I^n \setminus Q^n), \\ \bar{e}^c_i - \bar{e}^n_i = 1 - 1 = 0, & j_i \in Q^c \cap Q^n, \\ \bar{e}^c_i - \bar{e}^n_i = 0 - 0 = 0, & j_i \in (I^c \setminus Q^c) \cap (I^n \setminus Q^n), \\ \bar{e}^c_i - \bar{e}^n_i = 0 - 1 = -1, & j_i \in (I^c \setminus Q^c) \cap Q^n. \end{cases}$$

For $s \in J$ it holds that $\bar{a}_s = B^{-1}a_s$, that is, $a_s = B\bar{a}_s$. The pivot that leads from B^c to B^n must have the following column entering the basis.

$$a_s = B^c\bar{a}^c_s = \left(a^c_{j_1}, \ldots, a^c_{j_m}\right)\bar{a}^c_s =$$

$$= \sum_{i\in M} \bar{a}^c_{is}a^c_{j_i} = \sum_{j\in Q^c\cap J^n} a_j + \sum_{j\in Q^c\cap(I^n\setminus Q^n)} a_j - \sum_{j\in(I^c\setminus Q^c)\cap Q^n} a_j = \sum_{j\in J^c\cap Q^n} a_j = a^y$$

Hence, letting y enter the basis will yield a non-degenerate pivot-on-one leading to a unimodular basis B^n for the augmented problem. According to Lemma 1, the extreme point associated with this basis is integral.

Lemma 3, with $S = J^c \cap Q^n$, yields how x^n can be reconstructed in terms of original variables. ■

In the remainder of this section we discuss the effects that the introduction of a surrogate column will have on the properties of the restricted master problem. By its construction, the augmented problem is also a set partitioning problem. The dual problem of SPP^{LP}_N is

$$\max \quad \sum_{i\in M} e_i u_i \quad \text{s.t.} \quad c_j - \sum_{i\in M} a_{ij}u_i \geq 0, \quad j \in N.$$

Introducing the primal variable y, adds the surrogate constraint

$$\sum_{j\in J^c\cap Q^n} (c_j - \sum_{i\in M} a_{ij}u_i) \geq 0$$

to the dual problem. With y being a basic variable, equality holds in this constraint and the sum of the reduced costs of the variables $j \in J^c \cap Q^n$ is

zero. This can of course hold without each and one of these reduced costs being zero, and the consequences of this will be discussed further.

Set partitioning problems are typically highly degenerate, and the introduction of a surrogate column raises the degree of degeneracy; hence the extreme point $(\hat{x}, \hat{y})^n$ is likely to be highly degenerate, and therefore associated with several bases. These bases are associated with different dual extreme points, and for all of these, the dual surrogate constraint is fulfilled with equality, but the fulfillment of the original dual constraints can differ, that is, the values $\bar{c}_j^n$, $j \in J^c \cap Q^n$, can be both positive and negative, and this can of course be the case even if $(\hat{x}, \hat{y})^n$ is an optimal solution.

The above line of reasoning motivates that the original columns contained in the surrogate column should, if possible, be pivoted into the basis. It is probable that most of these pivots will be degenerate, which motivates the use of the degenerate all-integer pivots, introduced in Definition 3.

Since the cost of the surrogate column is the sum of the costs of the columns that it comprises, there are no incentives not to use the surrogate column. One way to prioritise the original columns over a surrogate column is to slightly penalise the latter. If more than one surrogate column is present, a ranking can be created by letting this penalty be proportional to the number of columns contained in the surrogate column, that is, by letting

$$c^y = |J^c \cap Q^n|\epsilon + \sum_{j \in J^c \cap Q^n} c_j,$$

where $\epsilon > 0$ is a value, smaller than anything to which it is compared.

If a surrogate column leaves the basis, it is no longer needed and can be removed from the problem. If, on the other hand, a surrogate column is still in the basis when the algorithm terminates, the solution can be translated into the original variables using the result in Lemma 3.

4.2 Optimality condition and over-generation

In a traditional linear programming column generation setting, each restricted master problem is solved to optimality, so that the column generation subproblem is provided with a dual feasible solution for the restricted problem. Because of this, the sign of the reduced cost of a generated column can solely be used as an indicator for optimality also in the complete master problem. Further, a generated column can be already present in the restricted master problem if and only if this problem already contains an optimal solution to the complete problem.

When applying column generation in an all-integer pivoting setting, the situation is quite different. Characteristic for this setting is that a linear pro-

gramming optimal basis for the restricted master problem SPP_N^{LP} is typically not available. The column generation subproblem using the dual variables of a non-optimal basis might then very well produce a column already present in the restricted master problem, and adding this column does of course not give any progress. Further, this might happen both in the situation when the current restricted master problem contains an optimal solution to the complete problem and when this is not the case.

To be able to solve the problem $\text{SPP}_{\mathcal{N}}$ to optimality by using an all-integer column generation approach, two essential questions therefore remain to be answered: "When is the restricted master problem ensured to contain an optimal solution to the complete master problem and how can it be ensured that it eventually will?"

The purpose of this section is to provide an optimality condition tailored for the all-integer column generation setting; a more generic version of this optimality condition is introduced in Rönnberg [15]. Also described in this section is the concept of over-generation of columns, which is a strategy for eventually complying with the presented optimality condition.

4.2.1 Optimality condition

The optimality conditions presented here can be used to determine when the restricted master problem SPP_N^{LP} is sufficiently large to be able to also give an optimal solution to the complete master problem, given any basis B and associated dual solution u.

To apply the result in Rönnberg [15], some new quantities need to be calculated. Let $\bar{c}_k = \min\{\bar{c}_j : \ j \in \mathcal{N} \setminus N\}$. (If the given basis for SPP_N^{LP} is not dual feasible, the reduced cost $\bar{c}_k$ might be non-optimal in the column generation problem [CG], and it can therefore be more cumbersome to calculate; see Section 4.2.2.) We denote the lower bound on the optimal objective value of SPP_N by $\underline{z}_R$ under the condition that $x_k = 1$ shall hold, and by $\underline{z}$ if $x_k = 0$ shall hold. To derive such bounds, we let $X \subseteq \{0,1\}^{|\mathcal{N}|}$ be a set containing all feasible solutions to $\text{SPP}_{\mathcal{N}}$ and let $Y \subseteq [0,1]^n$ be a set containing the orthogonal projection of X onto $\{0,1\}^n$. Further, assuming that assigning $x_k = 1$ imposes a restriction on the possible values of the variables in SPP_N, let the set R be chosen such that the following implication holds.

$$(x_j)_{j\in\mathcal{N}} \in X \cap \{(x_j)_{j\in\mathcal{N}} : \ x_k = 1\} \ \Rightarrow \ (x_j)_{j\in N} \in R \subseteq [0,1]^n$$

The lower bounds then become

$$\underline{z} = \sum_{i\in M} u_i b_i + \min\left\{\sum_{j\in N} \bar{c}_j x_j \ : \ (x_j)_{j\in N} \in Y\right\}$$

and

$$\underline{z}_R = \sum_{i \in M} u_i b_i + \min \left\{ \sum_{j \in N} \bar{c}_j x_j \ : \ (x_j)_{j \in N} \in Y \cap R \right\}.$$

Further introduce their difference, $\Delta = \underline{z}_R - \underline{z}$.

Another quantity needed in the optimality condition is an upper bound on the optimal objective value of $\mathrm{SPP}_{\mathcal{N}}$, denoted $\bar{z}$. A natural choice is to use the current solution to SPP_N.

The optimality condition then is as follows; for a more thorough motivation and a proof the reader is referred to [15].

Theorem 4 *If* $\bar{c}_k \geq 0$ *and*

$$\bar{c}_k + \Delta \geq \bar{z} - \underline{z},$$

hold, then an optimal solution to SPP_N *is also an optimal solution to* $SPP_{\mathcal{N}}$.

The optimality condition of the theorem is alternatively expressed on the form $\bar{c}_k \geq \bar{z} - \underline{z}_R$; which of the forms that is easiest interpreted depends on the setting, as illustrated by the examples below. If assigning $x_k = 1$ imposes no clear restriction on the possible values of the variables in SPP_N, the only possible choice is $R = [0,1]^n$, in which case $Y \cap R = Y$, $\underline{z}_R = \underline{z}$, and $\Delta = 0$ hold, so that the condition reduces to $\bar{c}_k \geq \bar{z} - \underline{z}$.

There might of course be different ways to obtain lower bounds, depending on the specific application under consideration and the amount of computations that can be allowed. Presented below are two examples of how to obtain lower bounds.

Example 1 *Let the set* X *be defined by a cardinality constraint on the number of variables that can take the value one in a solution, that is,*

$$X = \left\{ (x_j)_{j \in \mathcal{N}} \in \{0,1\}^{\mathcal{N}} : \ \sum_{j \in \mathcal{N}} x_j \leq L \right\}.$$

It follows directly from the set partitioning constraints that one may choose $L = m$*, but often the application at hand can facilitate a tighter choice.*

It is then natural to choose Y *as the orthogonal projection of* X *onto the set* $\{0,1\}^N$*, that is,*

$$Y = \left\{ (x_j)_{j \in N} \in \{0,1\}^{N} : \ \sum_{j \in N} x_j \leq L \right\},$$

and the following restriction imposed by assigning $x_k = 1$*,*

$$R = \left\{ (x_j)_{j \in N} \in \{0,1\}^N : \sum_{j \in N} x_j \leq L - 1 \right\}.$$

If the variables in SPP_N *are sorted such that* $\bar{c}_1 \leq \bar{c}_2 \leq \ldots \leq \bar{c}_n$*, the quantities to be calculated are*

$$\underline{z} = \sum_{i \in M} u_i e_i + \sum_{j=1}^{L} \bar{c}_j, \quad \underline{z}_R = \sum_{i \in M} u_i e_i + \sum_{j=1}^{L-1} \bar{c}_j, \quad \textit{and} \quad \Delta = -\min\{0, \bar{c}_L\},$$

and the optimality condition of Theorem 4 becomes $\bar{c}_k - \min\{0, \bar{c}_L\} \geq \bar{z} - \underline{z}$*.*

A case similar to the above example is when the variables of the problem $\text{SPP}_{\mathcal{N}}$ are partitioned in groups and the variables within each group is included in a convexity constraint. In this case the convexity constraints can similarly be used to define the sets X and R.

Another example, that is likely to yield a tighter bound but which requires some more calculations than the previous example, is as follows.

Example 2 *Let the set* X *be obtained from the original feasible set by replacing the set partitioning constraints with set packing constraints and let the set* Y *be obtained from of the orthogonal projection of* X *onto* $\{0,1\}^N$ *by relaxing the integrality restrictions on the variables. Then, a restriction imposed by assigning* $x_k = 1$ *is*

$$R = \left\{ (x_j)_{j \in N} \in [0,1]^N : \sum_{j \in N} a_{ij} x_j \leq e_i - a_{ik}, \ i \in M \right\}.$$

Expressing the optimality condition as $\bar{c}_k + \underline{z}_R \geq \bar{z}$ *and using that* $Y \cap R = R$ *holds, we obtain the optimality condition*

$$\bar{c}_k + \sum_{i \in M} u_i e_i + \min \left\{ \sum_{j \in N} \bar{c}_j x_j \ : \ (x_j)_{j \in N} \in Y \cap R \right\} \geq \bar{z}.$$

In order to check its fulfillment, a linear program thus needs to be solved.

4.2.2 Over-generation

To ensure progress of the all-integer approach, even though a column generated by the subproblem [CG] might already be included in the restricted

master problem, one possibility is to apply *over-generation of columns*, which means that the restricted master problem is augmented with a set of columns instead of just an optimal column. Over-generation can also be used to determine if it is guaranteed that the restricted master problem is sufficient for producing an optimal solution to $\text{SPP}_{\mathcal{N}}$, or to ensure that all columns needed to solve the problem $\text{SPP}_{\mathcal{N}}$ can eventually be found.

Definition 5 *Let the integer $K \geq 2$. The strategy of finding the K best solutions to the column generation subproblem* [CG] *and including them all in the restricted master problem is referred to as* over-generation. *With* exhaustive over-generation *we refer to the strategy of choosing the value of K so large that the optimality condition of Theorem 4 becomes fulfilled.*

Clearly, over-generation strategies can alternatively be defined in terms of finding all ε-optimal solutions to the subproblem [CG], for some $\varepsilon > 0$. Exhaustive over-generation is obtained for the choice $\varepsilon = \bar{z} - \underline{z} - \Delta - \bar{c}_p$, with the reduced cost $\bar{c}_p$ given by the column generation subproblem [CG].

If the basis in the restricted master problem is not dual feasible, the reduced cost $\bar{c}_k = \min\{\bar{c}_j : \ j \in \mathcal{N} \setminus N\}$, used in the optimality condition in Theorem 4, is not obtained automatically from the subproblem [CG], since its solution might violate the the requirement $k \in \mathcal{N} \setminus N$. This requirement can then be made fulfilled through over-generation of solutions to [CG].

The origin of the over-generation strategy can be found in a paper written by Sweeney and Murphy [17] in the late seventies. The computational results presented in their paper might no longer be of much interest, but the strategy introduced there has been applied as a vital component in a method presented in a more recent paper by Baldacci et al. [2] in 2008, where they successfully have solved vehicle routing problems. A theoretical justification for the over-generation principle can be found in Larsson and Patriksson [9], where it is however mainly discussed in a Lagrangian relaxation context.

The number of columns generated in an exhaustive over-generation, which guarantees that the restricted master problem SPP_N^{LP} can give an optimal solution to the set partitioning problem $\text{SPP}_{\mathcal{N}}$, depends on the quality of a lower bound. There might be several ways to obtain such a bound, but the tighter it is, the fewer columns will be required to be added to the restricted master problem. In practice, one needs to balance how much effort is put into the finding of a high quality lower bound and the over-generation of columns respectively.

It is not likely to be efficient to try to generate all columns with a reduced cost up to $\bar{z} - \underline{z} - \Delta$ if the current integer solution to SPP_N^{LP} is known to be far from optimal in $\text{SPP}_{\mathcal{N}}$, because the information contained in the dual

variables will then probably not be a reliable guide for which columns are useful in finding an optimal solution to $\text{SPP}_{\mathcal{N}}$. Later in the search, with a near-optimal integer solution to $\text{SPP}_{\mathcal{N}}$ and a basis with a near-feasible dual solution, the information contained in the dual variables should improve.

4.3 A transformation of the subproblem

When preparing this paper, some preliminary computational experiments were made for the generalised assignment problem in order to illustrate the practical behaviour of the core components of the all-integer column generation concept. These experiments revealed a risk of numerical difficulties in the column generation subproblem due to ill-conditioning of B; as new columns entered the basis the condition number increased until the subproblems could no longer be solved (by CPLEX 12.2) and the algorithm had to be terminated prematurely. At a first glance the matrix B looks harmless since it comprises set partitioning master columns, with zeros and ones only, and it is typically also sparse, but when the condition number becomes high (of the order 10^{16} for the instances used below) B^{-1} becomes dense with elements that range from very low to very high values (of the order from -10^6 to 10^7 for the instances used below), which makes the subproblem challenging.

This section begins with introducing the transformation used to overcome these numerical difficulties. Here, the transformation is made for the generalised assignment problem, but it can be done in the same manner for any subproblem formulated as a linear integer program. Thereafter we present the implementation (made in C++) which consists only of all-integer column generation, using CPLEX 12.2 for solving the subproblems.

4.3.1 The generalised assignment problem

The generalised assignment problem (GAP) is to assign n jobs to m agents at minimal cost. Each item should be assigned to one agent and the capacities of the agents should be respected. Let the capacity of agent i be denoted by b_i, and let p_{ij} and q_{ij} denote the cost and the claim of capacity respectively, when item j is assigned to agent i, $i = 1, \ldots, m$, $j = 1, \ldots, n$.

A formulation of GAP that can be used in a column generation setting is as follows, which yields a set partitioning master problem and knapsack subproblems. Let K_i be an index set for the feasible assignments of jobs to agent $i = 1, \ldots, m$ and define the variables

$$y_i^k = \begin{cases} 1, & \text{if assignment } k \text{ of agent } i \text{ is used,} \\ 0, & \text{otherwise,} \end{cases} \qquad k = 1, \ldots, K_i,\ i = 1, \ldots, m.$$

A feasible assignment $k = 1, \ldots, K_i$ for an agent $i = 1, \ldots, m$ is represented by the master problem column

$$x_{ij}^k = \begin{cases} 1, \text{ if item } j \text{ is assigned to agent } i \text{ in assignment } k, \ j = 1, \ldots, n, \\ 0, \text{ if item } j \text{ is not assigned to agent } i \text{ in assignment } k, \ j = 1, \ldots, n, \\ 1, \text{ if } j = n + i \text{ holds}, \ j = n + 1, \ldots, n + m \\ 0, \text{ if } j \neq n + i \text{ holds}, \ j = n + 1, \ldots, n + m, \end{cases}$$

where the elements x_{ij}^k, $j = n + 1, \ldots, n + m$, are introduced to simplify the notation by keeping track of which columns that belong to which agent i.

The master problem is

$$\begin{aligned} \text{[MP: SPP]} \qquad z^* \ = \ & \min \ \sum_{i=1}^{m} \sum_{k=1}^{K_i} \left(\sum_{j=1}^{n} p_{ij} x_{ij}^k \right) y_i^k \\ & \text{s.t.} \quad \sum_{i=1}^{m} \sum_{k=1}^{K_i} x_{ij}^k y_i^k = e_j, \quad j = 1, \ldots, n + m, \\ & \qquad y_i^k \in \{0, 1\}, \quad k = 1, \ldots, K_i, \ i = 1, \ldots, m, \end{aligned}$$

where $e_j = 1$, $j = 1, \ldots, n + m$. Denote the dual variables for the set partitioning constraints by u_j, $j = 1, \ldots, n + m$. The ordinary column generation subproblem can be formulated as follows, for each agent $i = 1, \ldots, m$.

$$\begin{aligned} \text{[SUB}_i\text{]} \qquad z^* \ = \ & \min \ \sum_{j=1}^{n} p_{ij} x_{ij} - \sum_{j=1}^{n+m} u_j x_{ij} \\ & \text{s.t.} \quad \sum_{j=1}^{n} q_{ij} x_{ij} \leq b_i, \\ & \qquad x_{ij} \in \{0, 1\}, \quad j = 1, \ldots, n, \\ & \qquad x_{ij} = 1, \text{ when } j = n + i, \ j = n + 1, \ldots, n + m \\ & \qquad x_{ij} = 0, \text{ when } j \neq n + i, \ j = n + 1, \ldots, n + m. \end{aligned}$$

The variables indexed $j = n + 1, \ldots, n + m$ are kept in the model, although they have fixed values, in order to simplify the transformation to be presented.

4.3.2 The transformation

In an all-integer column generation setting, the subproblem is tailored to produce a column that enables either a non-degenerate or a degenerate all-integer pivot by adding the side constraints introduced in Section 3.2. To

formulate these side constraints, B denotes the current master problem basis matrix, which is of size $(n+m)\times(n+m)$, and $\bar{e}_l$, $l=1,\ldots,n+m$, denote the updated right-hand-sides of the master problem.

Since the two types of side constraints are similar, we can form a common constraint for both the non-degenerate and degenerate cases and distinguish between them by changing the left-hand-side and the right-hand-side of the constraint. For each $i=1,\ldots,m$ this common constraint is stated as

$$lhs_l \le \sum_{j=1}^{n+m} \left(B^{-1}\right)_{lj} x_{ij} \le rhs_l,\ l=1,\ldots,n+m.$$

In the non-degenerate case, $lhs_l = -1+\bar{e}_l$ and $rhs_l = \bar{e}_l$, $l=1,\ldots,n+m$. In the degenerate case, two subproblems are formed for each r such that $\bar{e}_r = 0$, $r=1,\ldots,n+m$, by using

$$lhs_l = \begin{cases} \pm 1, & \text{for } l=r \\ -\infty, & \text{for } l\neq r \end{cases} \qquad l=1,\ldots,n+m,$$

$$rhs_l = \begin{cases} \pm 1, & \text{for } l=r \\ \infty, & \text{for } l\neq r \end{cases} \qquad l=1,\ldots,n+m.$$

The all-integer column generation subproblem for each agent $i=1,\ldots,m$ then is [SUB$_i$] augmented with the side constraint. The numerical difficulties in this formulation are overcome by eliminating B^{-1} from the subproblem using the following linear transformation. For each $i=1,\ldots,m$, let

$$s_l = \sum_{j=1}^{n+m} \left(B^{-1}\right)_{lj} x_{ij}, \quad l=1,\ldots,n+m,$$

be the variables in the transformed subproblem, which is constructed by substituting

$$x_{ij} = \sum_{l=1}^{n+m} B_{jl} s_l, \quad j=1,\ldots,n+m,$$

in the original side constrained subproblem. By making this substitution, B^{-1} is eliminated from the side constrained subproblem. Further, the binary restrictions on x_{ij}, $j=1,\ldots,n+m$, translate into requiring that the variables s_l, $l=1,\ldots,n+m$, shall take integer values (without sign restrictions). The

transformed side constrained column generation subproblem then becomes

$$
\begin{aligned}
[\text{AICG}_i] \quad \min\ \bar{c}_i \ = \ & \sum_{l=1}^{n+m} \left(\sum_{j=1}^{n} p_{ij} B_{jl} - (c_B)_l \right) s_l \\
\text{s.t.} \ & \sum_{l=1}^{n+m} \left(\sum_{j=1}^{n} q_{ij} B_{jl} \right) s_l \leq b_i, \\
& 0 \leq \sum_{l=1}^{n+m} B_{jl} s_l \leq 1, \quad j = 1, \ldots, n, \\
& \sum_{l=1}^{n+m} B_{jl} s_l = 1, \quad j = n+i, \ j = n+1, \ldots, n+m \\
& \sum_{l=1}^{n+m} B_{jl} s_l = 0, \quad j \neq n+i, \ j = n+1, \ldots, n+m \\
& lhs_l \leq s_l \leq rhs_l \text{ and integer}, \quad l = 1, \ldots, n+m,
\end{aligned}
$$

for each agent $i = 1, \ldots, m$.

4.3.3 Experiments on the generalised assignmnet problem

This section presents the results of a preliminary implementation made for illustrating the behaviour of core components of the all-integer column generation concept for the generalised assignment problem. As test instances we used the five gap 9 problems in the OR-library [4]; these are the largest of the most sparse instances, with 10 agents and 30 jobs. (Worth noting is that a drawback of using the generalised assignment problem for illustrating all-integer column generation is that the master problem is in this case more dense that in a typical applications of column generation.)

The implementation is based on making all-integer pivots on columns generated by solving AICG_i. The search is started from a purely artificial basis, for which the columns have been assigned the cost zero. A new column is generated and pivoted on by using the schema presented in Figure 1, and the algorithm is terminated after a predetermined number of iterations, or if it is not possible to perform any all-integer pivot.

When solving AICG_i, the agent and, in the degenerate case, the row are chosen at random. The algorithm was set to run 10,000 iterations for each of the five instances, without any special effort spent on tuning the parameters of the algorithm. The result is shown in Figure 2 which displays, for each iteration, the deviation in objective value between the current best solution and a known optimal one. The performance observed is like expected.

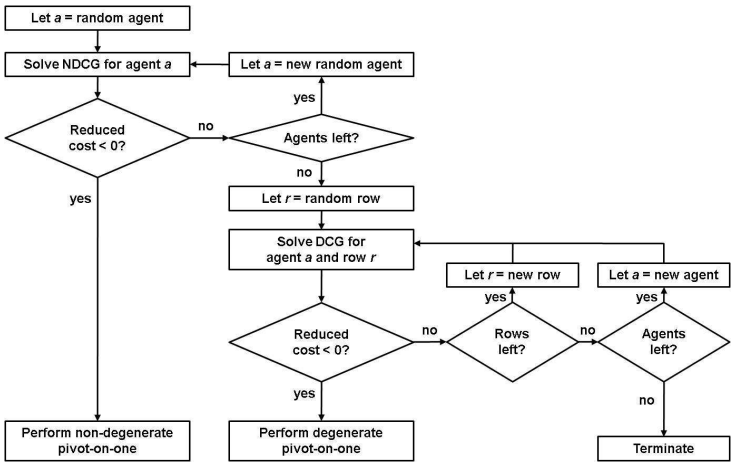

Figure 1: An overview of the implementation made for illustrating the core components of all-integer column generation.

In the search for a column for agent i, the problem AICG_i needs to be solved at most $1 + 2n$ times; once for the non-degenerate case and in the degenerate case twice for each degenerate row. (There are $n + m$ master problem rows, but since there are m different subproblems, at least m rows must be non-degenerate.) The strategy of consecutively solving these subproblems until a profitable column is found is feasible (but not necessarily efficient) for a small scale problem, but for a large scale problem more specialised strategies need to be designed.

5 Concluding remarks

The basic principles of all-integer column generation for set partitioning problems and the extensions presented are summarized in Figure 3 together with possible interactions between the components of the methodology. The strategies for solving and augmenting the restricted master problem can be combined to create both heuristic and exact methods.

The key characteristics of the all-integer column generation approach are the same as those of combinatorial metaheuristics, in that all successively found solutions are integral and feasible. Since the integrality requirements are maintained, the search can be interrupted at any time with a current best solution that is integral and feasible. The price to be paid for preserving integrality is that the all-integer pivots only serve as a local search method, and that, compared to traditional branch-and-price, the column generation subproblems are more complex.

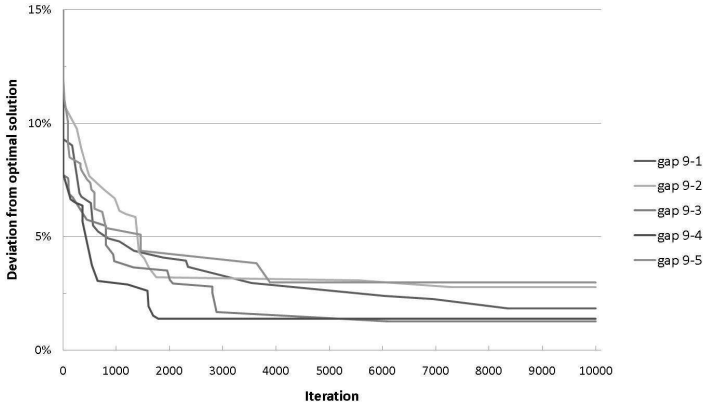

Figure 2: The chart shows how much the current best objective value deviates from the optimal one during the course of the algorithm.

Worth noting is that the surrogate column construction allows the use of any available and efficient method for finding improved integer solutions to the restricted master problem. In combination with for example a heuristic, the surrogate column pivots therefore enables a flexibility in the design of a specific implementation.

For the overall method to be exact, it must be guaranteed that an optimal solution to $\text{SPP}_{\mathcal{N}}$ will eventually be contained in SPP_N, for example by ensuring that the optimality condition in Theorem 4 become fulfilled through over-generation of columns, and that an optimal solution to SPP_N is eventually found; further, if degenerate all-integer pivots are used, cycling must be prevented. We do not know of any anti-cycling rule for the all-integer pivots, but the use of surrogate column pivots serves as a safeguard that prevents cycling implicitly. Further, it is not sufficient to know when an optimal solution to the complete master problem is contained in its restriction; an optimal solution must also be found, and this can be achieved by applying any suitable integer programming strategy. One way to guarantee optimality in SPP_N would be to use the branching strategy tailored to the integral simplex method that is presented by Thompson [19].

The example given in Section 4.3.3 serves as an illustration of how the framework presented can be applied and also introduces a transformation which we believe to be very important from a computational point of view. The computational results are merely used to illustrate the behaviour of the core components of the method and show that they behave as desired. It remains for future work to make a proper implementation of a method within our framework.

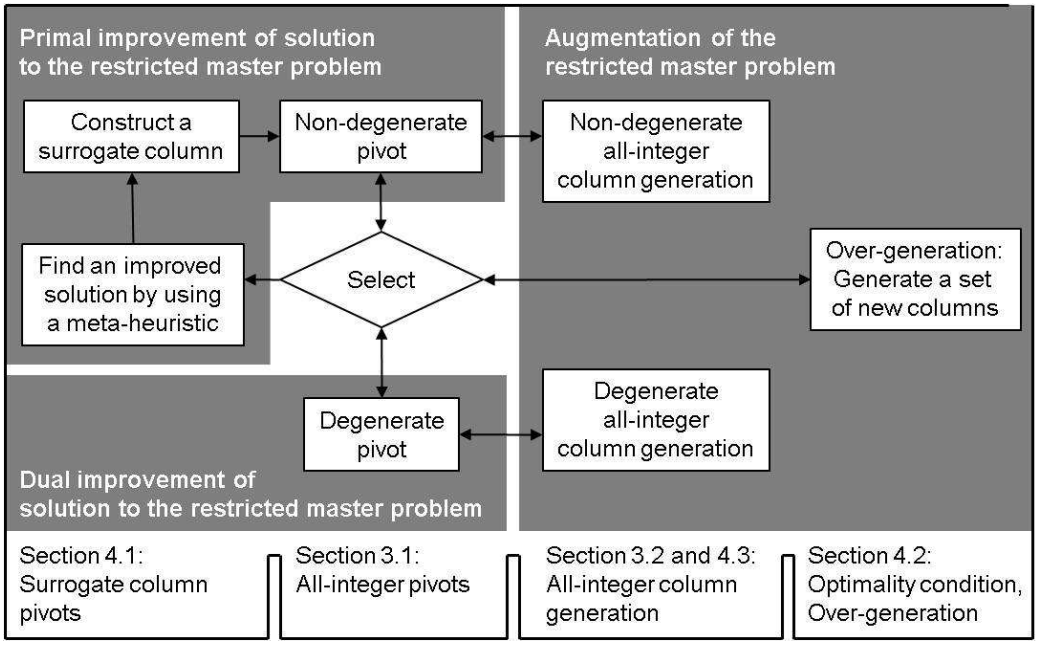

Figure 3: An outline of the components of the all-integer column generation approach.

We believe that the all-integer column generation approach has its main potential as a metaheuristic scheme for sparse large-scale set partitioning problems. In such a metaheuristic use of the approach, several feasible and successively improved solutions should be provided during the course of the solution process. The actual performance of a method within our framework will depend on a number of factors. Critical questions to address are how efficiently the column generation subproblems can be solved in the application at hand and the performance of the all-integer pivot search, as well as on the efficiency of the auxiliary method for finding improved integer solutions.

Acknowledgements

The authors were financially supported by a grant from the Swedish Research Council (grant no. 621-2005-2874). We also thank the referees whose valuable comments inspired major changes of the presentation of the paper.

References

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

[2] R. Baldacci, N. Christofides, and A. Mingozzi. An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming, series A*, 115:351–385, 2008.

[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. E. Beasley. OR-library, January 2011. URL `http://people.brunel.ac.uk/$\sim$mastjjb/jeb/orlib/gapinfo.html`.

[5] G. W. Brown and T. C. Koopmans. Computational suggestions for maximizing a linear function subject to linear inequalities. In T. C. Koopmans, editor, *Activity Analysis of Production and Allocation*, Monograph / Cowles commission for research in economics, pages 377–380, New York, NY, 1951. John Wiley & Sons.

[6] H. A. Eiselt and C.-L. Sandblom. External pivoting in the simplex algorithm. *Statistica Neerlandica*, 39:327–341, 1985.

[7] M. Gamache, F. Soumis, G. Marquis, and J. Desrosiers. A column generation approach for large-scale aircrew rostering problems. *Operations Research*, 47:247–263, 1999.

[8] C. Joncour, S. Michel, R. Sadykov, D. Sverdlov, and F. Vanderbeck. Column generation based primal heuristics. *Electronic Notes in Discrete Mathematics*, 36:695–702, 2010.

[9] T. Larsson and M. Patriksson. Global optimality conditions for discrete and nonconvex optimization—With applications to Lagrangian heuristics and column generation. *Operations Research*, 54:436–453, 2006.

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

[11] Metaheuristic Search by Column Generation, January 2013. URL `http://searchcol.dps.uminho.pt/publications.htm`.

[12] K. G. Murty. *Linear programming.* John Wiley & Sons, New York, NY, 1983.

[13] K. G. Murty and Y. Fathi. A feasible direction method for linear programming. *Operation Research Letters*, 3:121–127, 1984.

[14] E. Rönnberg. *Methods and applications in integer programming: All-integer column generation and nurse scheduling.* Licentiate thesis, Linköping University, Department of Mathematics, 2008. URL `http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-15143`.

[15] E. Rönnberg. *Contributions within two topics in integer programming: nurse scheduling and column generation.* PhD dissertation, Linköping University, Department of Mathematics, 2012. URL `http://liu.diva-portal.org/smash/record.jsf?searchId=1$\&$pid$=$diva2:512202$\&$rvn$=$3`.

[16] E. Rönnberg and T. Larsson. Column generation in the integral simplex method. *European Journal of Operational Research*, 192:333–342, 2009.

[17] D. J. Sweeney and R. A. Murphy. A method of decomposition for integer programs. *Operations Research*, 27:1128–1141, 1979.

[18] H. A. Taha. *Integer Programming: Theory, Applications, and Computations.* Academic Press, New York, NY, 1975.

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

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

[21] F. Vanderbeck. Branching in branch-and-price: a generic scheme. *Mathematical Programming*, 130:249–294, 2011.

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

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

[24] A. Zaghrouti, F. Soumis, and I. El Hallaoui. Integral simplex using decomposition for the set partitioning problem. *Unpublished manuscript*, 2013.