GERAD

GROUPE D'ÉTUDES ET DE RECHERCHE
EN ANALYSE DES DÉCISIONS

**Les Cahiers du GERAD**

*G-2014-14*
*March 2014*
*Revised: July 2015*

**CITATION ORIGINALE / ORIGINAL CITATION**

*Groiez, M., Desaulniers, G., Marcotte, O. Valid inequalities and separation algorithms for the set partitioning problem. INFOR 52(4), 185-196 (2014).*

**GERAD** HEC Montréal
3000, ch. de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

**Tél. : 514 340-6053**
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

**Les Cahiers du GERAD** ISSN: 0711–2440

# Valid inequalities and separation algorithms for the set partitioning problem

M. Groiez, G. Desaulniers,
O. Marcotte

G–2014–14

March 2014
Revised: July 2015



**GERAD** HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

**Tél. : 514 340-6053**
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

# Valid inequalities and separation algorithms for the set partitioning problem

**Mounira Groiez** [a]

**Guy Desaulniers** [a]

**Odile Marcotte** [b]

[a] *GERAD & Département de mathématiques et de génie industriel, Polytechnqiue Montréal, Montréal (Québec) Canada, H3C 3A7*

[b] *GERAD & Département d'informatique, Université du Québec à Montréal, Montréal (Québec) Canada, H3C 3P8*

mounira.groiez@gerad.ca
guy.desaulniers@gerad.ca
odile.marcotte@gerad.ca

**March 2014**
**Revised: July 2015**

**Les Cahiers du GERAD**
**G–2014–14**

**Abstract:** In this article we investigate some strategies for solving set partitioning problems (SPP), in particular the gains in computational efficiency that can be obtained by introducing cutting planes based on some rank-1 Chvátal-Gomory inequalities and clique inequalities. We show that for many instances of the SPP, the introduction of some of these cutting planes into the standard SPP model enables a commercial solver such as CPLEX to compute optimal solutions more efficiently.

**Key Words:** Set partitioning, cutting planes, rank-1 Chvátal-Gomory inequalities, clique, separation algorithm.

**Résumé :** Dans cet article, nous étudions des stratégies pour résoudre le problème de partitionnement d'ensemble (PPE), en particulier les gains en efficacité qui peuvent être obtenus grâce à des plans de coupure basés sur des inégalités de Chvátal-Gomory de rang 1 et des inégalités de cliques. Nous montrons que pour beaucoup d'exemplaires du PPE, l'introduction de certains de ces plans de coupure dans la formulation standard du PPE permet à un logiciel commercial tel que CPLEX de calculer plus rapidement des solutions optimales.

**Mots clés :** Partitionnement d'ensemble, plans coupants, inégalités de rang 1, clique, algorithme de séparation.

**Acknowledgments:** The authors are very grateful to NSERC for its support through the discovery grant of Guy Desaulniers. They also wish to thank the two referees for their helpful comments and suggestions.

# 1 Introduction

Given the set $U = \{1, 2, \ldots, m\}$ and a family $\mathcal{F}$ of subsets of $U$ denoted $U_1$, $U_2$, ..., $U_n$, the *set partitioning problem* (or *partitioning problem* for short) consists of finding a subfamily of $\mathcal{F}$ that is a partition of $U$, i.e., a subset $J$ of $\{1, 2, \ldots, n\}$ such that $U$ equals $\bigcup_{j \in J} U_j$ and $U_j \cap U_k$ is empty for any two distinct indices $j$ and $k$ in $J$. The sets in $\mathcal{F}$ can be given explicitly (as lists of elements in $U$) or implicitly (as sets verifying some condition). For instance one could define $\mathcal{F}$ as the family of all paths from the vertex $s$ to the vertex $t$ in some graph $G$. The partitioning problem we have introduced is one of *feasibility*. If we assign weights $w_j$ to the subsets in $\mathcal{F}$, we can define an optimization problem (also called a partitioning problem) by requesting an *optimal partition*, i.e., a subfamily of $\mathcal{F}$ whose total weight is minimal.

Partitioning problems are closely related to packing and covering problems, which play an important role in combinatorial optimization, from a theoretical as well as a practical point of view (see Cornuéjols 2001). For instance vertex colouring can be viewed as a partitioning or covering problem; bin packing and many vehicle or crew scheduling problems can be modelled as partitioning problems. On the other hand, many fundamental problems in combinatorial optimization are packing or partitioning problems: the maximum-weight matching problem is a packing problem while the minimum-weight perfect matching problem is a partitioning problem. Thus when attempting to solve partitioning problems, it is natural to use tools developed for solving well-known combinatorial problems such as the stable set and matching problems. In this article we focus on one of these tools, the generation of cutting planes.

It is well known that the set partitioning problem (SPP) is NP-complete. Actually the special case of this problem where $|U|$ is a multiple of 3 and each $U_i$ contains three elements was proved to be NP-complete by Karp in a seminal paper (see Karp 1972). Generally speaking the SPP has received less attention in the operations research literature than the packing and covering problems. The article by Balas and Padberg (1976) is a survey of theoretical results and solution methods for partitioning and covering problems. In particular its authors discuss the relationship between the set partitioning problem and other combinatorial optimization problems (node packing, set packing, clique covering), facets of the set packing polytope, and algorithms for solving the set partitioning problem (implicit enumeration, cutting plane methodes, column generation algorithms). Note that Padberg (1973) had already proved that valid inequalities for the SPP based on cliques (see the next paragraph) actually define facets of the SPP polytope.

Hoffman and Padberg (1993) present a theoretical and practical study of the set partitioning problem (within the context of the crew scheduling problem, which can be modelled as an SPP). Their theoretical study includes a discussion of the relationship between the facets of the SPP polytope and the stable set problem for a graph related to the SPP. More precisely, they define an intersection graph $G$ as follows: the vertex set of $G$ is $\{1, 2, \ldots, n\}$ and its edge set $\{(i, j) \mid U_i \cap U_j = \emptyset\}$. Then they derive inequalities for the SPP that are based on cliques, odd cycles, and complements of odd cycles in the graph $G$. On the practical side, they describe in detail a branch-and-cut approach that includes model preprocessing and cutting plane identification and generation (including the lifting of their coefficients).

Chu and Beasley (1998) propose a genetic algorithm and Thompson (2002) an integral simplex algorithm for solving the SPP. Elhallaoui et al. (2005) consider the set partitioning formulation of some vehicle routing and crew scheduling problems; they propose a method for aggregating set partitioning constraints within the framework of a column generation algorithm. Lewis et al. (2008) formulate the SPP as a quadratic programming problem and describe a tabu-based heuristic for solving it. In many applications (e.g., vehicle routing or crew scheduling applications), the model contains both set partitioning constraints and so-called supplementary constraints. The literature contains few data sets corresponding to "pure" SPPs, i.e., SPPs without supplementary constraints. Such data sets can be found in Hoffman and Padberg (1993) and Lewis et al. (2008).

As mentioned above, Hoffman and Padberg (1993) drew their inspiration from the stable set problem (also known in the literature as the independent set or vertex packing problem). In the present article we derive inequalities of a different sort, similar to the inequalities defining the matching polytope. The latter inequalities were introduced by Edmonds (1965) in a classical article; they are a special case of the so-called rank-1 Chvátal-Gomory inequalities (see Chvátal 1973). Padberg and Rao (1982) give a polynomial-time

separation algorithm for detecting the violated odd-set inequalities of the matching polytope and Grötschel, Lovasz, and Schrijver (1993) a polynomial-time separation algorithm for the odd-hole inequalities of the stable set polytope (see also Nemhauser and Sigismondi 1992).

Caprara and Fischetti (1996) prove that for a general integer program, the separation of rank-1 Chvátal-Gomory inequalities is NP-complete (see also Jepsen et al. 2008), while Caprara et al. (2000) show that maximally violated mod-$k$ cuts can be separated in polynomial time. In the same vein we mention the article by Andreello et al., which present a computational study of the introduction of $\{0, \frac{1}{2}\}$-rank-1 cuts into a Branch-and-Cut framework. In the rest of our article we use the phrase "rank-1 inequalities" to mean "rank-1 Chvátal-Gomory inequalities" and "rank-1 cutting planes" to mean "rank-1 Chvátal-Gomory cutting planes."

The main contribution of the present article is the investigation of the added value (so to speak) of $\{0, \frac{1}{2}\}$-rank-1 cutting planes (see definition below), since the value of these cutting planes for the partitioning problem does not seem to have been investigated before. Also we propose integer programming formulations for separating clique-based inequalities and $\{0, \frac{1}{2}\}$-rank-1 inequalities. The article is organized as follows. In Section 2 we describe $\{0, \frac{1}{2}\}$-rank-1 inequalities for the set partitioning problem and a method for separating them. In Section 3 we describe a method for separating clique-based inequalities. In Section 4 we present our algorithm. In Section 5 we discuss our experimental results and in Section 6 we outline avenues for future work.

## 2 Separating $\{0, \frac{1}{2}\}$-rank-1 inequalities

In order to formulate the set partitioning problem as an integer program, we introduce the matrix $A = (a_{ij})$ whose columns are the incidence vectors of the $U_j$. More precisely, for $i$ in $\{1, 2, \ldots, m\}$ and $j$ in $\{1, 2, \ldots, n\}$, we define $A$ in such a way that $a_{ij}$ equals 1 if the element $i$ belongs to $U_j$ and $a_{ij}$ equals 0 otherwise. For $j$ in $\{1, 2, \ldots, n\}$, we let $x_j$ denote a binary variable that is equal to 1 if and only if the set $U_j$ belongs to the partition. The SPP can then be formulated as the following integer program, denoted also by $(SPP)$, where $\mathbf{1}$ is a column vector in which every entry equals 1.

$$\min \quad \sum_{j=1}^{n} c_j x_j$$

subject to

$$\begin{aligned} Ax &= \mathbf{1} \\ x_j &\in \{0,1\} \qquad \forall j = 1, 2, \ldots, n. \end{aligned}$$

Let $P_{int}$ denote the convex hull of the feasible solutions of the partitioning problem. Then a rank-1 valid inequality for $P_{int}$ (see Chvátal 1973) is obtained by choosing nonnegative rational numbers $\lambda_i$ for $i$ in $\{1, 2, \ldots, m\}$ and writing

$$\sum_{j=1}^{n} \left\lfloor \sum_{i=1}^{m} \lambda_i a_{ij} \right\rfloor x_j \leq \left\lfloor \sum_{i=1}^{m} \lambda_i \right\rfloor. \tag{1}$$

Note that in the original article, there is no $\lfloor r \rfloor$ symbol (denoting the greatest integer at most $r$) in the left-hand side of this relation. If a system includes nonnegativity constraints, however, the above inequality is valid. If every $\lambda_i$ equals 0 or 1/2, this inequality is called a $\{0, \frac{1}{2}\}$-rank-1 inequality. Clearly each feasible integral solution of $(SPP)$ satisfies any $\{0, \frac{1}{2}\}$-rank-1 inequality, which is therefore "valid." Of course, if we assume that each $U_j$ contains exactly two elements, the partitioning problem reduces to a minimum-cost perfect matching problem; then the convex hull of feasible integral solutions is described by the original inequalities (the "degree constraints") and the $\{0, \frac{1}{2}\}$-rank-1 inequalities (see Edmonds 1965).

Let us consider a given $\{0, \frac{1}{2}\}$-rank-1 valid inequality and define $S$ as $\{i \mid \lambda_i > 0\}$, $J_0(S)$ as $\{j \mid \sum_{i\in S} a_{ij}$ is even$\}$, and $J_1(S)$ as $J \backslash J_0(S)$. Then Inequality (1) can be rewritten as

$$\sum_{j\in J_0(S)} \left(\frac{\sum_{i\in S} a_{ij}}{2}\right) x_j + \sum_{j\in J_1(S)} \left(\frac{\sum_{i\in S} a_{ij} - 1}{2}\right) x_j \leq \left\lfloor \frac{|S|}{2} \right\rfloor. \tag{2}$$

This inequality may "cut" a fractional solution of the linear relaxation of $(SPP)$ only if $|S|$ is an odd number. Thus to find an inequality of the form (2) that is violated by the current solution of the linear relaxation of $(SPP)$ (denoted by $x^*$), it suffices to find an odd set $S$ and a subset $J_1(S)$ of $\{1, 2, \ldots, n\}$ such that the following relations hold.

$$|S| \text{ is odd} \tag{3}$$

$$j \in J_1(S) \text{ if and only if } \sum_{i\in S} a_{ij} \text{ is odd} \tag{4}$$

$$\sum_{j\in J_0(S)} \left(\frac{\sum_{i\in S} a_{ij}}{2}\right) x^*_j + \sum_{j\in J_1(S)} \left(\frac{\sum_{i\in S} a_{ij} - 1}{2}\right) x^*_j > \left\lfloor \frac{|S|}{2} \right\rfloor \tag{5}$$

As mentioned in the introduction, the separation of $\{0, \frac{1}{2}\}$-rank-1 inequalities is an NP-complete problem. In this article we propose a formulation of the separation problem as an integer program (see Groiez et al. 2013 for a similar formulation). The conditions (3), (4), and (5), however, are cumbersome. We now replace (2) by an equivalent (and simpler) inequality.

**Proposition 2.1** *If $|S|$ is odd, Inequality (2) is equivalent to*

$$\sum_{j\in J_1(S)} x_j \geq 1. \tag{6}$$

**Proof.** Adding all the inequalities of the system $Ax = 1$ that correspond to rows in $S$ and dividing by 2 yields

$$\sum_{j\in J_0(S)} \left(\frac{\sum_{i\in S} a_{ij}}{2}\right) x_j + \sum_{j\in J_1(S)} \left(\frac{\sum_{i\in S} a_{ij}}{2}\right) x_j = \frac{|S|}{2}. \tag{7}$$

Subtracting Inequality (2) from this inequality yields

$$\sum_{j\in J_1(S)} \frac{x_j}{2} \geq \frac{1}{2}, \tag{8}$$

hence the conclusion of the proposition. □

Proposition 2.1 is a translation, in formal terms, of the following simple observation: any partition of an odd set $S$ must contain an odd subset of $S$. In our case the partition of $S$ is induced by a partition $\mathcal{F}$ of $U$ and the observation can be rephrased as follows: $\mathcal{F}$ must contain a subset $U_j$ such that $|S \cap U_j|$ is odd.

To find a violated inequality of the form (6), one needs to find an odd set $S$ and a subset $J_1(S)$ of $\{1, 2, \ldots, n\}$ such that the following relations hold.

$$|S| \text{ is odd} \tag{9}$$

$$j \in J_1(S) \text{ if and only if } \sum_{i\in S} a_{ij} \text{ is odd} \tag{10}$$

$$\sum_{j\in J_1(S)} x^*_j < 1. \tag{11}$$

We introduce an integer linear program, called the *auxiliary integer program*, for separating Inequalities (6). The objective of this program is to minimize $\sum_{j\in J_1(S)} x^*_j$, since there exists a violated inequality of the form (6) if and only if the minimum value of $\sum_{j\in J_1(S)} x^*_j$ is smaller than 1. We let $w_i$ denote a binary variable that equals 1 if and only if $i$ belongs to $S$ and $z_j$ a binary variable that equals 1 if and only if $j$ belongs to $J_1(S)$, i.e., $j$ is such that $\sum_{i\in S} a_{ij}$ is odd. Finally let $k$ denote a variable representing the value $\lfloor |S|/2 \rfloor$ and $\ell_j$ (for all $j$) a variable whose value equals $\left\lfloor \left(\sum_{i\in S} a_{ij}\right)/2 \right\rfloor$. Here is the integer linear program, which is similar to that found in Koster et al. (2009).

$$\min \quad \sum_{j=1}^{n} x^*_j z_j$$

subject to

$$\sum_{i=1}^{m} w_i = 2k+1 \tag{12}$$

$$\sum_{i=1}^{m} a_{ij} w_i - z_j - 2\ell_j = 0 \qquad \forall j = 1,2,\ldots,n \tag{13}$$

$$w_i \in \{0,1\} \qquad \forall i \in \{1,2,\ldots,m\}$$

$$k \in \mathbb{Z}$$

$$z_j \in \{0,1\}, \ell_j \in \mathbb{Z} \qquad \forall j = 1,2,\ldots,n$$

The constraint (12), combined with the integrality requirements, enforces the definition of $S$ (i.e., $S$ is an odd set). The constraints (13) enforce the definition of $J_1(S)$, the $z_j$, and the $\ell_j$ at the same time. If $x^*_j$ equals 0, it is not necessary to include the index $j$ into the auxiliary program, and if both $x^*_j$ and $z_j$ equal 1, the optimal value of the auxiliary program will be at least 1. Hence in practice we can define $J'$ as the index set $\{j \mid 0 < x^*_j < 1\}$ and solve the above auxiliary integer program after replacing "$\forall j = 1,2,\ldots,n$" by "$\forall j \in J'$."

Also in practice it may be necessary to introduce a parameter $k_{max}$ that is an upper bound on $k$ and add to the auxiliary integer program the constraint $1 \le k \le k_{max}$. Then any solution returned by the auxiliary program will represent an odd subset of cardinality at most $2k_{max}+1$. The upper bound on $k$ may be needed to ensure that the solution time of the auxiliary integer program is not too large.

Note also that any feasible solution of this program whose value is less than 1 gives rise to a valid inequality violated by the current solution. More precisely, if we have $\sum_{j\in J'} x^*_j z_j < 1$ and define $S$ as $\{i \mid w_i = 1\}$, $J'_1$ as $\{j \mid z_j = 1\}$, and $J'_0$ as

$$\left\{ j \mid x^*_j = 0 \text{ and } \sum\nolimits_{i\in S} a_{ij} \text{ is odd} \right\},$$

then we can define $J_1(S)$ as $J'_1 \cup J'_0$ and include the constraint $\sum_{j\in J_1(S)} x_j \ge 1$ into the model to "cut" the current solution, i.e., $x^*$. Therefore when we run CPLEX or some other software in order to solve the auxiliary program, we can generate as many cutting planes as feasible solutions of value less than 1 found by CPLEX. In practice we impose a limit on the running time of the auxiliary program because solving it to optimality is time-consuming.

## 3 Separating clique inequalities

The valid inequalities introduced in the last section are similar to the blossom inequalities of the matching polytope, inasmuch as they are $\{0, \frac{1}{2}\}$-rank-1 valid inequalities. The analogy between the perfect matching problem and $(SPP)$ arises if we consider the rows of the matrix $A$ as vertices in an undirected graph and the columns of $A$ as "edges." Of course a column of $A$ may contain more than two ones, which makes $(SPP)$ much more difficult than the perfect matching problem. We can also draw an analogy between $(SPP)$ and

the stable set problem by defining the graph $G = (V, E)$, where $V$ is the set of column indices of the matrix $A$ and $E$ is the set

$$\{jk \mid a_{ij} = a_{ik} = 1 \text{ for some } i\}.$$

In other words, $G$ is the graph describing the "conflicts" between the columns of $A$. If $\{U_j\}_{j\in J}$ is a partition of $U$, then $J$ is a stable set in $G$. It is easy to verify that for any clique $C$ in $G$, the inequality $\sum_{j\in C} x_j \leq 1$ is valid for $(SPP)$; indeed Balas and Padberg (1976) prove that this inequality is a facet of the polytope associated with $(SPP)$.

The problem of finding a violated clique inequality can be formulated as a maximum-weight clique problem. The latter is one of the fundamental problems in combinatorial optimization and has several formulations as a mathematical programming problem; many algorithms have been proposed to solve it (see Bomze et al. 1999 for a survey). In order to attempt finding many violated clique inequalities, we formulate a program to find a maximum-weight clique of cardinality $k$ (where $k$ is a constant). Let $B$ denote the vertex-edge incidence matrix of $G$, i.e., $b_{je}$ equals 1 if the vertex $j$ belongs to the edge $e$ and 0 otherwise. We also let $y_j$ denote a binary variable that equals 1 if and only if $j$ belongs to the clique and $w_e$ a binary variable that equals 1 if and only if $e$ is an edge of the clique. Finally we let $k$ denote the cardinality of the clique, which we assume to be fixed. The objective of the following integer program will be greater than 1 if and only if a clique inequality is violated by the solution $x^*$ (for some clique of cardinality $k$).

$$\max \quad \sum_{j=1}^{n} x_j^* y_j$$

subject to

$$\sum_{j=1}^{n} y_j = k \tag{14}$$

$$\sum_{e\in E} w_e = \frac{k(k-1)}{2} \tag{15}$$

$$Bw = (k-1)y \tag{16}$$

$$y_j \in \{0,1\} \qquad \forall j = 1, 2, \ldots, n$$

$$w_e \in \{0,1\} \qquad \forall e \in E$$

The relations (14) and (15) enforce the constraints that the number of vertices equals $k$ and the number of edges equals $k(k-1)/2$, respectively. A typical relation in the system $Bw = (k-1)y$ is of the form $B_j w = (k-1)y_j$ and enforces the constraint that the degree of the vertex $j$ equals $k-1$.

Note that for a fixed $k$, any feasible solution of the above program whose value is greater than 1 gives rise to a valid inequality violated by the current solution of the linear relaxation. If a feasible solution of the auxiliary program (denoted by $y$) verifies $\sum_{j=1}^{n} x_j^* y_j > 1$, we define $C$ as $\{j \in J \mid y_j = 1\}$ and add the inequality $\sum_{j\in C} x_j \leq 1$ to the model in order to "cut" the current solution of the linear relaxation (i.e., $x^*$). As in the case of $\{0, \frac{1}{2}\}$-rank-1 inequalities, we impose a time limit on the auxiliary program. If no cutting plane is found for a clique of cardinality $k$, we do not attempt to solve the auxiliary program with the parameter $k+1$, because we surmise that it will be too difficult to solve. In practice we solve the auxiliary integer program for $k = 4$, $k = 5$, and $k = 6$ (note that the case $k = 3$ corresponds to a special case of the $\{0, \frac{1}{2}\}$-rank-1 inequalities).

# 4 The algorithm

The algorithm we propose is an "enhanced" version of a commercial solver (CPLEX, in this instance), where we add cutting planes at the root node. More precisely the integer program considered at the root node, i.e., $(SPP)$ itself, is improved by the inclusion of cutting planes. Here is a summary of our algorithm.

Algorithm 1 uses CPLEX to solve the linear relaxation of $(SPP)$ and find a feasible (integral) solution of $(SPP)$. We then add as many $\{0, \frac{1}{2}\}$-rank-1 cutting planes and clique-based cutting planes as we can, solve

**Algorithm 1** An algorithm for solving $(SPP)$

1: Use CPLEX to solve the linear relaxation of $(SPP)$ and find an integral feasible solution (or determine that none exists).
2: Attempt to find violated $\{0, \frac{1}{2}\}$-rank-1 and clique-based inequalities.
3: **while** the separation algorithms have found at least one violated inequality **do**
4: Add the violated inequalities to the model.
5: Solve the new linear relaxation.
6: Attempt to find violated $\{0, \frac{1}{2}\}$-rank-1 and clique-based inequalities.
7: **end while**
8: Continue solving $(SPP)$ by using the CPLEX branch-and-bound algorithm.

the linear relaxation once more, and repeat. When we cannot find violated inequalities any more, we use the CPLEX branch-and-bound algorithm to solve the current integer program. Note that in the first step of Algorithm 1, which consists of finding an integral feasible solution or concluding that none exists, we do not introduce our cutting planes. This would involve replacing the heuristic procedure of CPLEX with our own.

## 5 Experimental results

In order to test our algorithm, we used data sets provided by Hoffman and Padberg (1993) and Lewis et al. (2008). (We mentioned in Section 1 that few data sets could be found in the literature.) We carried out our tests with the default version of CPLEX 12.4.0.0 on a machine with two Opteron 250 processors, 2.4 GHz, and a 16G memory. We did not use the parallel version of CPLEX since many observations have demonstrated that the one-thread version is more efficient. We also used the "preprocessing" feature of CPLEX for all our experiments.

Out of the 55 instances in Hoffman and Padgerg (1993), 52 could be solved by CPLEX in less than 20 seconds; therefore, because there is not much to gain for these 52 instances, we kept the three remaining instances only (sppaa01, appnw04, and appus01). We will discuss our experiments with these three instances at the end of the current section. Lewis et al. (2008) propose a generator of random instances that generates an SPP instance of a given size and a given density. We used this generator to produce 5 instances in each of 11 categories, where each category is defined by $m$ (the number of elements in $U$), $n$ (the number of subsets $U_j$ in $\mathcal{F}$), and its density (i.e., the proportion of nonzero entries in the matrix $A$). The characteristics of the instances are displayed in Table 1. Note that for the generated instances, the last two columns contain average values: for example the average solution value for the 5 instances in the Pb0 group equals 409.2. We now describe the results of our experiments on the randomly generated instances.

Table 1: Characteristics of the instances

| Instance | Nb. constr. | Nb. var. | Density | Lin. relax. | Opt. value |
|---|---|---|---|---|---|
| sppaa01 | 823 | 8904 | 1.0 | 55535.44 | 56137.0 |
| sppnw04 | 36 | 87482 | 20.2 | 16310.67 | 16862.0 |
| sppus01 | 145 | 1053137 | 9.1 | 9963.07 | 10036.0 |
| Pb0 | 300 | 1300 | 3.0 | 285.34 | 409.2 |
| Pb1 | 250 | 650 | 4.0 | 1548.04 | 2485.6 |
| Pb2 | 200 | 500 | 6.0 | 1146.14 | 2510.2 |
| Pb3 | 1500 | 1650 | 12.0 | 43751.01 | 131688.0 |
| Pb4 | 300 | 2300 | 16.0 | 17.35 | 3526.8 |
| Pb5 | 1500 | 2000 | 23.0 | 13285.78 | 116002.0 |
| Pb6 | 2300 | 300 | 34.0 | 7.05 | 4009.8 |
| Pb7 | 400 | 1400 | 58.0 | 7.34 | 3335.6 |
| Pb8 | 300 | 1300 | 59.0 | 5.76 | 2434.0 |
| Pb9 | 200 | 1400 | 62.0 | 3.67 | 1509.4 |
| Pb10 | 1300 | 300 | 82.0 | 3.84 | 1061.2 |

Since CPLEX allows one to use certain types of cutting planes and in particular the so-called zero-half cutting planes (which are actually $\{0, \frac{1}{2}\}$-rank-1 inequalities), we decided to start by comparing three "versions" of CPLEX: the version with no cutting planes whatsoever, the default version (with a "moderate" search for cutting planes), and the version with an aggressive search of cutting planes. The results of our comparison are presented as a collection of performance profiles in Figure 1 for the random instances generated by the generator of Lewis et al. We recall briefly how those performance profiles are obtained (see Dolan and Moré 2002). Let $t_{is}$ denote the solution time for the $i$th instance when Strategy $s$ is used, and $r_{is}$ (the *performance ratio*) be defined as

$$r_{is} = \frac{t_{is}}{\min_s\{t_{is}\}}.$$

For any real number $\tau$ such that $\tau \geq 1.0$ holds, we consider the set of instances $\{i \mid r_{is} \leq \tau\}$, i.e., those instances on which Strategy $s$ "behaves well" (with respect to $\tau$). Finally the *performance* of Strategy $s$ is defined as the function $\rho_s$:

$$\rho_s(\tau) = \frac{|\{i \mid r_{is} \leq \tau\}|}{N},$$

where $N$ denotes the number of instances. The performance profile of Strategy $s$ is the graph of the function $\rho_s$.

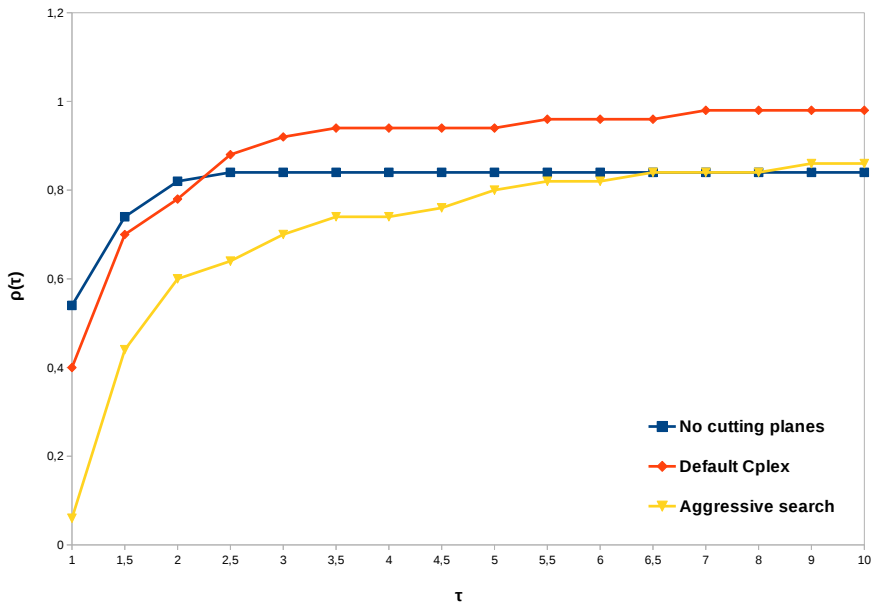


Figure 1: Comparison of various versions of CPLEX

The experiments summarized in Figure 1 show that the best version of CPLEX is the default version. Hence we have used that version when comparing our strategies with CPLEX (later in this section). It is interesting to note that the default version of CPLEX does not find many zero-half inequalities: actually, in 80% of the cases, it does not find any such inequality. We have also observed that CPLEX did not generate any clique-based cutting plane for the high-density instances.

We now turn to our own strategies. Note that we added cutting planes at the root node only, because finding them at each node of the branch-and-bound tree is expensive and also because CPLEX itself looks for cutting planes at the root node only. We now describe our strategies, the first three of which use the default version of CPLEX but also look for other cutting planes (based on the inequalities described in Sections 2 and 3, respectively).

1. Use the default version of CPLEX and look for violated clique inequalities.
2. Use the default version of CPLEX and look for violated $\{0, \frac{1}{2}\}$-rank-1 inequalities.

3. Use the default version of CPLEX and look for both violated clique inequalities and violated $\{0, \frac{1}{2}\}$-rank-1 inequalities.
4. Disable the use of CPLEX cutting planes and look for violated clique inequalities and violated $\{0, \frac{1}{2}\}$-rank-1 inequalities.

For detecting violated inequalities (of either sort), we solve an auxiliary program with the following parameter settings: HEURFREQ = 1, MIPEMPHASIS = 4, RINSHEUR = 1, and POLISHAFTERINTSOL = 1. This choice allows us to find as many feasible solutions as possible. Note that when one uses "polishing", it is difficult to assess the usefulness of the other parameters. We have also imposed an upper bound (10 seconds) on the solution time of any auxiliary program.

Our results, summarized in Figure 2, show that the best strategy is Strategy 4, which does not use any of CPLEX cutting planes but uses violated clique inequalities and violated $\{0, \frac{1}{2}\}$-rank-1 inequalities. Also we have observed that the number of violated clique inequalities is more or less the same in Strategies 1, 3, and 4, while the number of violated $\{0, \frac{1}{2}\}$-rank-1 inequalities is more or less the same in Strategies 2, 3, and 4. We are now ready to compare the best "CPLEX version" and the "best strategy." The comparison is illustrated in Figure 3. In general Strategy 4 has a better performance than the default version of CPLEX: this may be due to the fact that Strategy 4 generates more (and sometimes many more) violated $\{0, \frac{1}{2}\}$-rank-1 inequalities than the default version of CPLEX. On the other hand the default version of CPLEX finds more violated clique inequalities than Strategy 4 in certain cases. Finally we note that among the $\{0, \frac{1}{2}\}$-rank-1 inequalities found by Strategy 4, between 3% and 28% actually correspond to cliques of cardinality 3.

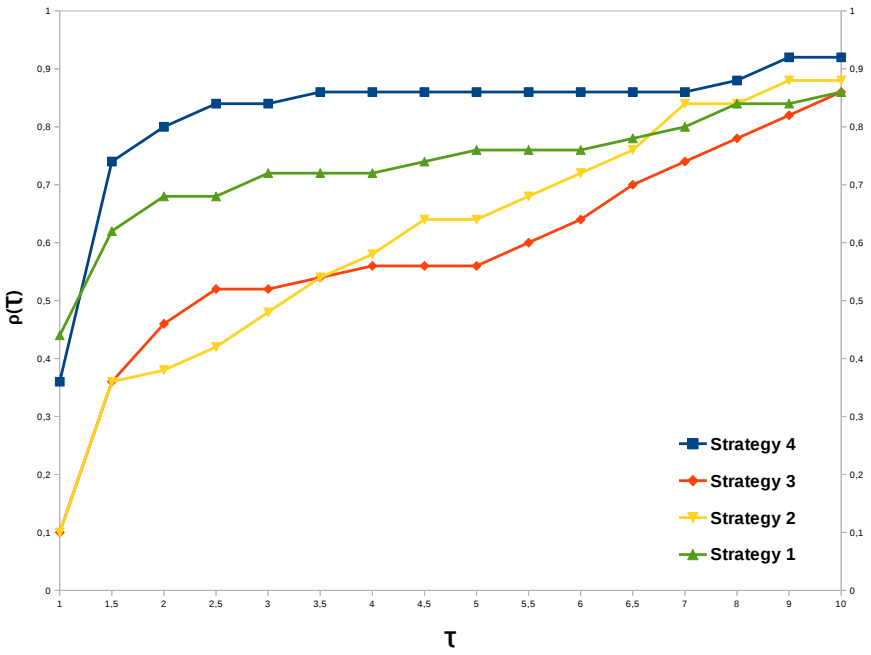


Figure 2: Performance profiles of the strategies

Figure 3 also illustrates the results of our experiments with Strategy 5, in which we use the auxiliary program of Section 2 to separate the $\{0, \frac{1}{2}\}$-rank-1 inequalities but use a greedy heuristic described in Hoffman and Padberg (1993) to separate the clique inequalities. Actually this heuristic finds cliques that are likely to yield violated clique inequalities and selects those that yield such inequalities. In 60% of cases all the cliques found by the greedy algorithm yield violated clique inequalities, and the greedy algorithm finds many more violated inequalities than either the default version of CPLEX or our algorithm (this is especially true in the case of high-density instances). Also the time spent separating violated clique inequalities by the greedy algorithm is at most 4% of the total solution time, except for the class Pb5 of instances, where the clique separation time is around 78% of the total solution time. On the other hand our algorithm for separating

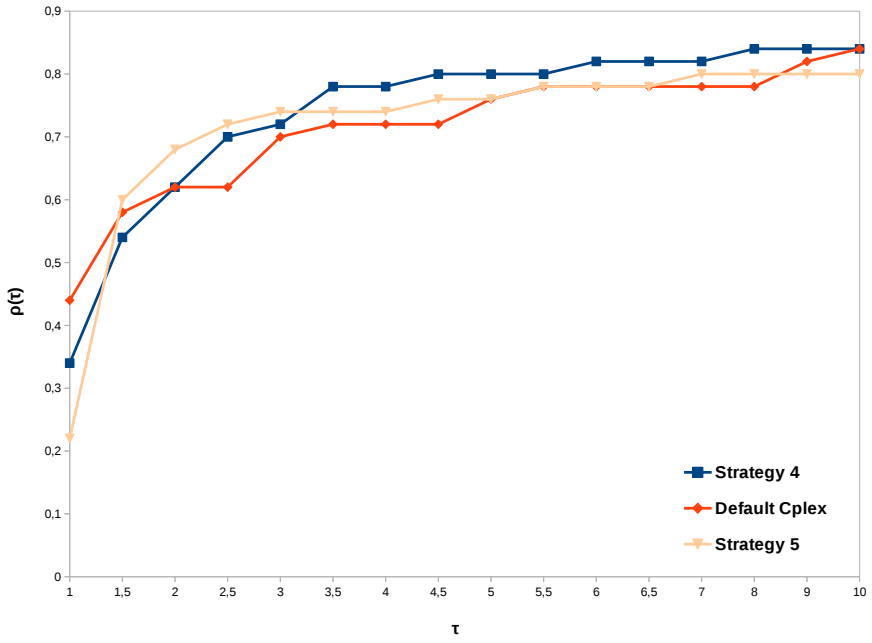


Figure 3: Performance profiles for all the random instances

violated clique inequalities (see Section 3) is at most 40% of the total solution time. In spite of this Strategy 4 is more efficient than either Strategy 5 or the default version of CPLEX.

It is reasonable to suspect that the performance of our algorithm is related to the instances density. We tested this hypothesis by splitting the pool of instances into those of high density (density greater than 16%) and those of low density (density at most 16%). Figure 4 shows that for the high-density instances, Strategy 4 outperforms both Strategy 5 and the default version of CPLEX. Figure 5 shows that for the low-density instances, the default version of CPLEX outperforms both Strategy 4 and Strategy 5. Note that one would

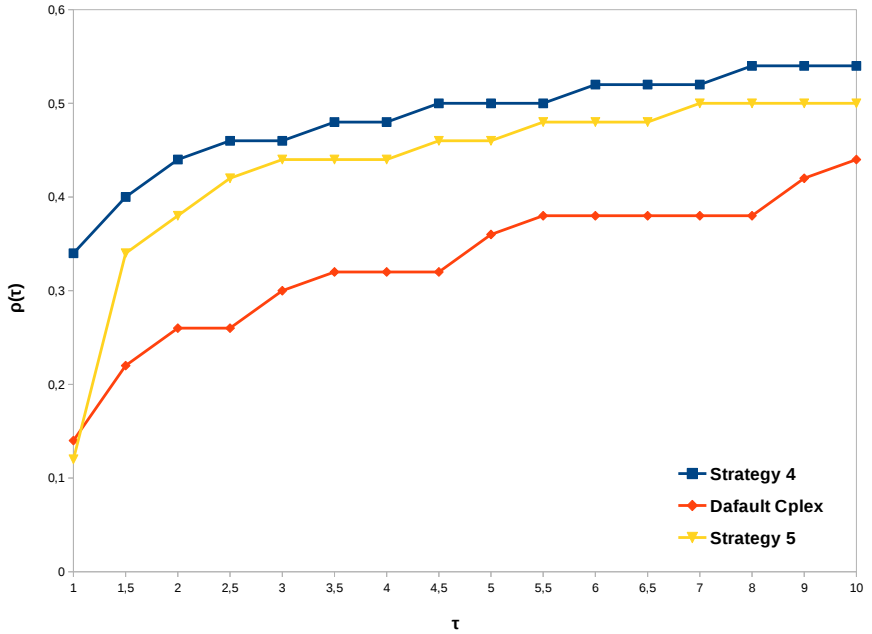


Figure 4: Performance profiles for the high-density random instances

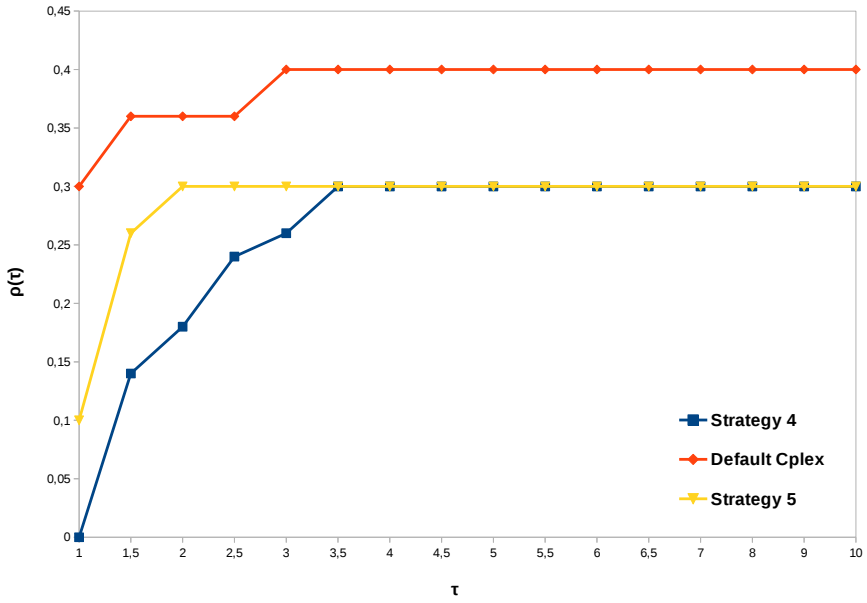


Figure 5: Performance profiles for the low-density random instances

expect a high-density instance to give rise to many violated clique inequalities, but that only the greedy algorithm succeeds in finding many inequalities of this type. For the instances in the classes Pb7, Pb8, Pb9, and Pb10, neither Strategy 4 nor the default version of CPLEX finds any clique of size at least 4.

In Table 2 we summarize the results for the randomly generated instances. For each class of problems (consisting of five instances) and each strategy (CPLEX default version, Strategy 4, and Strategy 5), we give the gap at the root node, the total number of nodes produced by CPLEX, and the total time spent by CPLEX to find the optimal solution or determine that the instance has no feasible solution. The gap is defined as $(UB - OPT) \times 100$, where $UB$ is the bound obtained at the root node after "tightening" the linear relaxation and $OPT$ is the optimal value of the integer program. The general conclusion is that Strategy 4 is the fastest option for dense instances, those in the classes Pb8, Pb9, and Pb10: this is due to the tightening at the root node, which reduces the gap significantly. The gap was also reduced in the case of the PB7 class, but this reduction did not result in a decrease in the total computing time. We observe that the policy of generating many violated clique inequalities is not necessarily the best one.

Table 2: Summary of results for the randomly generated instances

| | **Default version of CPLEX** | | | **Strategy 4** | | | **Strategy 5** | | |
|---|---|---|---|---|---|---|---|---|---|
| Class | Gap | Nodes | Time | Gap | Nodes | Time | Gap | Nodes | Time |
| Pb0 | 26.67 | 15738170.0 | 59734.2 | 26.30 | 13826980.6 | 50537.0 | 30.13 | 25218174.6 | 92445.1 |
| Pb1 | 37.72 | 80618.2 | 171.6 | 35.75 | 60888.0 | 248.0 | 36.05 | 53972.0 | 137.9 |
| Pb2 | 54.34 | 155153.6 | 235.8 | 52.52 | 145374.6 | 304.3 | 53.17 | 116166.8 | 189.0 |
| Pb3 | 66.78 | 1.0 | 1.6 | 66.73 | 11695.2 | 164.1 | 23.71 | 512.6 | 152.0 |
| Pb4 | 99.50 | 3290404.6 | 172318.1 | 99.49 | 8350817.4 | 172773.3 | 99.39 | 7964686.2 | 172660.0 |
| Pb5 | 88.48 | 256444.4 | 19721.9 | 88.48 | 163384.8 | 12118.5 | 73.27 | 12208.8 | 1629.0 |
| Pb6 | 99.82 | 2829735.2 | 138190.0 | 99.77 | 4040919.8 | 172280.8 | 99.71 | 4240498.2 | 172570.8 |
| Pb7 | 99.78 | 18648.6 | 1929.6 | 66.95 | 18746.2 | 2434.7 | 99.72 | 23104.8 | 2749.4 |
| Pb8 | 99.76 | 16240.8 | 1141.1 | 53.45 | 9110.6 | 663.4 | 99.57 | 14387.8 | 1160.9 |
| Pb9 | 99.76 | 23632.6 | 732.2 | 81.28 | 4858.2 | 200.9 | 99.72 | 13601.0 | 509.5 |
| Pb10 | 99.64 | 2022.4 | 595.1 | 68.98 | 708.6 | 156.1 | 99.51 | 2285.2 | 355.9 |

In Table 3 we give details on the cuts generated by our algorithm. We now explain the acronyms used to label the column tables.

- CL stands for the number of clique-based cutting planes found by the strategy (CPLEX default version, Strategy 4, or Strategy 5).
- ZH stands for the number of zero-half cutting planes found by the default version of CPLEX.
- GF stands for the number of Gomory fractional cutting planes found by the default version of CPLEX.
- Total refers to the total number of cutting planes found by any given strategy.
- K3 stands for the number of cliques of cardinality 3 found by our auxiliary integer program for $\{0, \frac{1}{2}\}$-rank-1 cuts (recall that the cutting plane arising from a clique of cardinality 3 is also a $\{0, \frac{1}{2}\}$-rank-1 cutting plane).
- OT ("other") denotes the number of $\{0, \frac{1}{2}\}$-rank-1 cutting planes found by our auxiliary integer program that are *not* of the form K3.
- TC denotes the time spent finding clique-based cutting planes.
- TR1 denotes the time spent finding $\{0, \frac{1}{2}\}$-rank-1 cutting planes.
- CF stands for the number of cliques found by the heuristic algorithm in Strategy 5.
- CV stands for the number of cliques giving rise to violated inequalities (among those found by the heuristic algorithm).

Table 3: Additional results for the randomly generated instances

| | **Default CPLEX** | | | | **Strategy 4** | | | | | | **Strategy 5** | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Class | CL | ZH | GF | Total | K3 | OT | CL | Total | TC | TR1 | K3 | OT | CF | CV | Total | TC | TR1 |
| Pb0 | 3.2 | 0.4 | 1.6 | 5.2 | 6.6 | 25.0 | 1.6 | 33.2 | 209.5 | 0.1 | 5.2 | 20.8 | 875.8 | 6.2 | 32.2 | 0.0 | 59.9 |
| Pb1 | 19.2 | 7.6 | 1.8 | 28.6 | 4.0 | 18.0 | 6.4 | 28.4 | 98.8 | 13.7 | 3.6 | 14.4 | 381.8 | 11.2 | 29.2 | 0.0 | 32.0 |
| Pb2 | 19.6 | 6.4 | 3.0 | 29.0 | 4.2 | 10.6 | 8.6 | 23.4 | 79.6 | 10.5 | 2.8 | 7.2 | 339.6 | 14.6 | 24.6 | 0.0 | 22.0 |
| Pb3 | 431.4 | 0.0 | 0.0 | 431.0 | 0.0 | 0.0 | 0.0 | 0.0 | 2.4 | 6.1 | 45.0 | 96.6 | 15176.4 | 15.2 | 156.8 | 6.1 | 139.9 |
| Pb4 | 2131.0 | 0.0 | 0.0 | 2131.0 | 0.0 | 0.0 | 0.0 | 0.0 | 30.4 | 0.0 | 0.0 | 0.0 | 959.4 | 959.4 | 959.4 | 5.1 | 10.0 |
| Pb5 | 0.0 | 0.0 | 8.4 | 8.4 | 0.0 | 0.0 | 0.0 | 0.0 | 24.6 | 0.1 | 0.0 | 0.0 | 81380.4 | 75.8 | 75.8 | 1268.0 | 20.3 |
| Pb6 | 8.8 | 0.0 | 9.8 | 18.6 | 0.0 | 0.0 | 0.0 | 0.0 | 0.6 | 0.0 | 0.0 | 0.0 | 236.0 | 236.0 | 236.0 | 8.0 | 10.0 |
| Pb7 | 0.0 | 0.0 | 22.8 | 22.8 | 0.0 | 1.6 | 0.0 | 1.6 | 4.0 | 0.9 | 0.0 | 1.2 | 391.0 | 391.0 | 392.2 | 40.2 | 10.4 |
| Pb8 | 0.0 | 0.0 | 12.0 | 12.0 | 0.6 | 19.2 | 0.0 | 19.8 | 2.5 | 11.0 | 0.2 | 2,8 | 293.8 | 293.8 | 296.8 | 11.4 | 10.1 |
| Pb9 | 0.0 | 0.0 | 13.2 | 13.2 | 2.2 | 19.8 | 0.0 | 22.0 | 1.2 | 30.9 | 0.0 | 3.0 | 195.2 | 195.2 | 198.2 | 1.7 | 10.0 |
| Pb10 | 0.0 | 0.0 | 31.8 | 31.8 | 0.0 | 27.6 | 0.0 | 27.6 | 1.9 | 67.7 | 0.0 | 2.2 | 292.6 | 292.6 | 294.8 | 11.0 | 11.2 |

It is interesting to note that Strategy 5 (or more precisedly the Hoffman-Padberg greedy heuristic) finds many more cutting planes than Strategy 4, especially in the case of the dense instances. Strategy 5, however, consumes more time than Strategy 4. Thus an increase in the number of cutting planes generated does not always lead to a more efficient strategy. Also our auxiliary integer program for finding clique-based cutting planes did not find any clique for the dense instances: this may be due to the fact that this auxiliary program is difficult to solve. Finally Tables 4 and 5 display the same kind of results for the three Hoffman-Padberg instances as were displayed for the random instances. We note that the introduction of our cutting planes does not enable one to solve these instances more efficiently than the default version of CPLEX.

Table 4: Summary of results for the Hoffman-Padberg instances

| | **Default version of CPLEX** | | | **Strategy 4** | | | **Strategy 5** | | |
|---|---|---|---|---|---|---|---|---|---|
| Class | Gap | Nodes | Time | Gap | Nodes | Time | Gap | Nodes | Time |
| sppaa01 | 1.07 | 258 | 6.6 | 1.01 | 523 | 83.4 | 0.99 | 349 | 41.9 |
| sppnw04 | 3.27 | 259 | 23.6 | 3.14 | 576 | 13.8 | 3.14 | 830 | 17.2 |
| sppus01 | 0.73 | 10 | 339.3 | 0.35 | 7 | 311.3 | 0.54 | 26 | 323.9 |

Table 5: Additional results for the Hoffman-Padberg instances

| | **Default CPLEX** | | | | **Strategy 4** | | | | | | **Strategy 5** | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Class | CL | ZH | GF | Total | K3 | OT | CL | Total | TC | TR1 | K3 | OT | CF | CV | Total | TC | TR1 |
| sppaa01 | 28 | 12 | 0 | 40 | 1 | 30 | 16 | 47 | 52.9 | 19.5 | 1 | 28 | 0 | 0 | 29 | 0.0 | 29.9 |
| sppnw04 | 30 | 0 | 0 | 30 | 0 | 1 | 1 | 2 | 0.0 | 4.7 | 0 | 1 | 0 | 0 | 1 | 0.0 | 4.7 |
| sppus01 | 16 | 2 | 0 | 18 | 0 | 9 | 6 | 15 | 4.7 | 11.0 | 0 | 9 | 0 | 0 | 9 | 0.1 | 11.0 |

# 6 Conclusion

In this article we propose to exploit certain classes of cutting planes in order to solve set partitioning problems. To find those cutting planes, we formulate and solve auxiliary integer programming problems. Introducing these cutting planes into the model at the root node and continuing the solution process with a commercial software (CPLEX, in this instance) enables us to solve the dense instances faster. For low-density instances, however, this approach is not very useful. In future work we would like to investigate the low-density instances and try to design fast algorithms (possiby heuristic ones) for computing cutting planes. We would also like to investigate the potential usefulness of $\{0, \frac{1}{2}\}$-rank-1 cuts for reducing the time spent finding the first feasible solution of a partitioning problem.

# References


Andreello, G., Caprara, A., Fischetti, M. (2007). Embedding $\{0, \frac{1}{2}\}$-Cuts in a Branch-and-Cut Framework: A Computational Study. INFORMS Journal on Computing, 19, 229–238.

Balas, E., Padberg, M.W. (1976). Set Partitioning : A Survey. SIAM Review, 18, 710–760.

Bomze, I.M., Budinich, M., Pardalos, P.M., Pelillo, M. (1999). The maximum clique problem. Handbook of Combinatorial Optimization, 4, 1–74, Kluwer Academic Publishers.

Caprara, A., Fischetti, M. (1996). $\{0, \frac{1}{2}\}$-Chvátal-Gomory Cuts. Mathematical Programming (A), 74, 221–235.

Caprara, A., Fishetti, M., Letchford, A.N. (2000). On the Separation of Maximally Violated mod-$k$ Cuts. Mathematical Programming, 87, 37–56.

Chu, P.C, Beasley, J.E. (1998). Constraint Handling in Genetic Algorithms: The Set Partitioning Problem. Journal of Heuristics, 11, 323–357.

Chvátal, V. (1973). Edmonds polytopes and a hierarchy of combinatorial problems. Discrete Mathematics, 4, 305–337.

Cornuéjols, G. (2001). Combinatorial Optimization: Packing and Covering. CBMS-NSF Regional Conference Series in Applied Mathematics, 74, SIAM.

Dolan, E.E., Moré, J.J. (2002). Benchmarking optimization software with performance profiles. Mathematical Programming, 91, 201–213.

Edmonds, J. (1965). Maximum Matching and a Polyhedron with 0-1 Vertices. Journal of Research of the National Bureau of Standards, 69B, 125–130.

Elhallaoui, I., Villeneuve, D., Soumis, F., Desaulniers, G. (2005). Dynamic Aggregation of Set-Partitioning Constraints in Column Generation. Operations Research, 53, 632–645.

Groiez, M., Desaulniers, G., Hadjar, A., Marcotte, O. (2013). Separating Valid Odd-Cycle and Odd-Set Inequalities for the Multiple Depot Vehicle Scheduling Problem. EURO Journal on Computational Optimization, 1, 283–312.

Grötschel, M., Lovász, L., Schrijver, A. (1993). Geometric Algorithms and Combinatorial Optimization, 2nd edition. Springer-Verlag Berlin Heidelberg.

Hoffman, K.L., Padberg, M. (1993). Solving Airline Crew Scheduling Problems by Branch-and-Cut. Management Science, 39, 657–682.

Jepsen, M., Petersen, B., Spoorendonk, S., Pisinger, D. (2008). Subset-Row Inequalities Applied to the Vehicle-Routing Problem with Time Windows. Operations Research, 2, 497–511.

Karp, R.M. (1972). Reducibility Among Combinatorial Problems. Complexity of Computer Computations (R.E. Miller and J.W. Thatcher, Eds.), 85–103.

Koster, A.M.C.A., Zymolka, A., Kutschka, M. (2009). Algorithms to Separate $\{0, \frac{1}{2}\}$-Chvátal-Gomory Cuts. Algorithmica, 55, 375–391.

Lewis, M., Kochenberger, G., Alidaee, B. (2008). A New Modeling and Solution Approach for the Set-Partitioning Problem. Computers and Operations Research, 35, 807–813.

Nemhauser, G.L., Sigismondi, G. (1992). A Strong Cutting Plane/Branch-and-Bound Algorithm for Node Packing. Journal of the Operational Research Society, 43, 443–457.

Padberg, M.W. (1973). On the facial structure of set packing polyhedra. Mathematical Programming, 5, 199–215.

Padberg, M.W., Rao, M.R. (1982). Odd Minimum Cut-Sets and $b$-matchings. Mathematics of Operations Research, 7, 67–80.

Thompson, G.L. (2002). An Integral Simplex Algorithm for Solving Combinatorial Optimization Problems. Computational Optimization and Applications, 22, 351–367.