Math. Program., Ser. B (2007) 110:3–20
DOI 10.1007/s10107-006-0054-8

FULL LENGTH PAPER

Optimizing over the first Chvátal closure

Matteo Fischetti · Andrea Lodi

Received: 28 July 2005 / Accepted: 25 June 2006 /
Published online: 15 December 2006

Abstract How difficult is, in practice, to optimize *exactly* over the first Chvátal closure of a generic ILP? Which fraction of the integrality gap can be closed this way, e.g., for some hard problems in the MIPLIB library? Can the first-closure optimization be useful as a research (off-line) tool to guess the structure of some relevant classes of inequalities, when a specific combinatorial problem is addressed? In this paper we give answers to the above questions, based on an extensive computational analysis. Our approach is to model the rank-1 Chvátal-Gomory separation problem, which is known to be NP-hard, through a MIP model, which is then solved through a general-purpose MIP solver. As far as we know, this approach was never implemented and evaluated computationally by previous authors, though it gives a very useful separation tool for general ILP problems. We report the optimal value over the first Chvátal closure for a set of ILP problems from MIPLIB 3.0 and 2003. We also report, for the first time, the optimal solution of a very hard instance from MIPLIB 2003, namely *nsrand-ipx*, obtained by using our cut separation procedure to preprocess the original ILP model. Finally, we describe a new class of ATSP facets found with the help of our separation procedure.

Work partially supported by MIUR, Italy, and by the EU projects *ADONET* (contract n. MRTN-CT-2003-504438) and *ARRIVAL* (contract no. FP6-021235-2).

M. Fischetti (✉)
DEI, University of Padova, via Gradenigo 6A 35131, Padova, Italy
e-mail: matteo.fischetti@unipd.it

A. Lodi
DEIS, University of Bologna, viale Risorgimento 2 40136, Bologna, Italy
e-mail: alodi@deis.unibo.it

Keywords Integer programs · Separation problems · Chvátal–Gomory cuts · Computational analysis

1 Introduction

We consider the integer linear program (ILP) problem

$$\min\{c^{\mathrm{T}}x : Ax \le b, x \ge 0 \text{ integer}\} \tag{1}$$

where A is a $m \times n$ matrix, $b \in \mathbb{R}^m$, and $c \in \mathbb{R}^n$, along with the two associated polyhedra:

$$P := \left\{x \in \mathbb{R}^n_+ : Ax \le b\right\} \tag{2}$$

$$P_I := \operatorname{conv}\left\{x \in Z^n_+ : Ax \le b\right\} = \operatorname{conv}(P \cap Z^n) \tag{3}$$

Unless explicitly stated, we assume that (A, b) is an integer matrix. Moreover, we assume that P_I is strictly contained in P, i.e., that P has fractional vertices.

A *Chvátal–Gomory* (*CG*) *cut* [15,9] is a valid inequality for P_I of the form

$$\lfloor u^{\mathrm{T}}A \rfloor x \le \lfloor u^{\mathrm{T}}b \rfloor\,, \tag{4}$$

where $u \in R^m_+$ is called the *CG multiplier vector*, and $\lfloor\cdot\rfloor$ denotes lower integer part. Note that CG cuts depend on P and not directly on P_I, i.e., different formulations $Ax \le b, x \ge 0$ of the same integer problem can produce different CG cuts. Inequality (4) is said to be of *rank 1* with respect to the formulation $Ax \le b, x \ge 0$; CG cuts of rank $h \ge 2$ are obtained recursively, starting with an enlarged system containing all CG cuts of rank less than h. It is known [19] that undominated CG cuts are obtained by using rational CG multipliers $u_i \in [0, 1)^m$ only, provided that (A, b) is integer.

The first *Chvátal closure* of P is defined as

$$P_1 := \left\{x \ge 0 : Ax \le b, \lfloor u^{\mathrm{T}}A \rfloor x \le \lfloor u^{\mathrm{T}}b \rfloor \text{ for all } u \in \mathbb{R}^m_+\right\} \tag{5}$$

A basic result, due to Chvátal [9], is that P_1 is indeed a polyhedron, i.e., a finite number of CG cuts suffice to define it.

Clearly, $P_I \subseteq P_1 \subseteq P$. It was shown by Gomory [15,16] that every fractional vertex x^* of P associated with a certain basis B (say) of (A, I) can be cut off by the CG cut in which u is chosen as the ith row of B^{-1}, where i is the row associated with any fractional component of x^*. Therefore, $P_1 \subset P$ in case $P \ne P_I$, i.e., P_1 gives a better approximation of P_I than P. In some cases, one has that $P_I = P_1$ as, e.g., for matching problems where undominated CG cuts correspond to the famous Edmonds' blossom inequalities [11,19]. Even in case $P_I \subset P_1$, however, we are interested in optimizing $c^{\mathrm{T}}x$ over P_1 in order to get a hopefully tight lower bound on the optimal value of the original ILP problem (1).

By the well-known equivalence between optimization and separation [19], we will address the *CG separation problem* in which we are given any point $x^* \in \mathbb{R}^n$ and we ask for a hyperplane separating x^* from P_1 (if any). Without loss of generality we can assume that $x^* \in P$, since all other points can be cut by simply enumerating the members of the original inequality system $Ax \leq b$, $x \geq 0$. Therefore, the separation problem we are actually interested in, called *CG-SEP* in the sequel, reads:

Definition 1 *(*CG-SEP) Given any point $x^* \in P$ find (if any) a CG cut that is violated by x^*, i.e., find $u \in \mathbb{R}^m_+$ such that $\lfloor u^{\mathrm{T}} A \rfloor x^* > \lfloor u^{\mathrm{T}} b \rfloor$, or prove that no such u exists.

It was proved by Eisenbrand [13] (through a reduction from the weighted binary clutter problem) that CG-SEP is NP-hard, so optimizing over P_1 also is.

It is worth observing that this result is not in contrast with the fact that any *vertex* of P can easily be cut by a CG cut read from the associated tableau. Indeed, the equivalence between optimization and separation relies on the use of the ellipsoid method, and this method cannot succeed by going from vertex to vertex. To be more specific, the convergence in a polynomial number of iterations of the ellipsoid method requires to consider a sequence of points x^* generated as centers of appropriate ellipsoids, so the point x^* to be cut off at any main iteration of the ellipsoid method cannot be assumed to have any special property such as, e.g., being integer, or being a vertex of P—just for the same reason that the convergence of binary search in a logarithmic number of steps requires to consider the middle point of the current interval at each step (considering any other point can affect the logarithmic convergence of the method). Therefore, in this context the "trick" of replacing x^* by a nearby vertex of P does not work, even if this operation can be performed in polynomial time, since the polynomial convergence of the overall ellipsoid method would be affected. On the other hand, the use of a cutting plane method based on the simplex (as opposed to the ellipsoid) method will guarantee that the point x^* to be cut at any iteration is a vertex of the current formulation, but this approach can generate an exponential number of CG cuts (some of which may even be of rank larger than 1), so this method cannot lead to an efficient approach for optimizing over P_1 (besides the fact that the simplex algorithm itself is non-polynomial).

In this paper we address the issue of evaluating the practical strength of P_1 in approximating P_I. Our approach is to model the rank-1 CG separation problem as a MIP model, which is then solved through a general-purpose MIP solver. As far as we know, this approach was never implemented (and evaluated computationally) by previous authors.[1] As our investigation is restricted to the study of the CG cuts, we will not consider other classes of stronger cuts such as, e.g., Gomory's mixed-integer cuts (or their equivalent *MIR* and *split cut* versions; see [19]), though these cuts may be very effective in practice. Indeed, in our

[1] The fact that CG separation can be modeled as a MIP can instead be attributed to *folklore* and is implicit, e.g., in the work of Caprara and Letchford [7] on the related class of split cuts.

view the heuristic inclusion in our framework of other families of cuts would hide the intrinsic value of CG cuts, hence making the overall picture less clear.

The paper is organized as follows. In Sect. 2 we describe our MIP model for CG separation. In Sect. 3 we investigate computationally the practical (and theoretical) benefits derived from using our separation tool. In particular, we report the optimal value over the first Chvátal closure for a set of problems from MIPLIB 3.0 [6] and MIPLIB 2003 [1]. We also report, for the first time, the optimal solution of a very hard instance, namely *nsrand-ipx*, obtained by using our CG separation procedure to preprocess the original ILP model. Finally, we describe a new class of ATSP facets found with the help of our CG separation procedure.

An extended abstract of the present paper appeared in the proceedings of the IPCO XI meeting [14]. Following our approach, other researchers addressed different types of closures for MIP problems; in particular, Balas and Saxena [4] considered the optimization over the split closure, whereas Dash, Günlük and Lodi [10] considered the optimization over the equivalent MIR closure [10]; finally, Bonami et al. [5] introduced the concept of *projected CG cuts* for mixed-integer problems, and investigated both theoretically and computationally the corresponding closure.

2 How?

The first question we address is how to solve the NP-hard CG-SEP problem.

2.1 A MIP model

Given the input point $x^* \geq 0$ to be separated, CG-SEP calls for a CG cut $\alpha^{\mathrm{T}} x \leq \alpha_0$ which is (maximally) violated by x^*, where $\alpha = \lfloor u^{\mathrm{T}} A \rfloor$ and $\alpha_0 = \lfloor u^{\mathrm{T}} b \rfloor$ for a certain $u \in \mathbb{R}_+$. A first CG-SEP model then reads:

$$\max \quad \alpha^{\mathrm{T}} x^* - \alpha_0 \tag{6}$$

$$\alpha_j \leq u^{\mathrm{T}} A_j \quad \text{for } j = 1, \ldots, n \tag{7}$$

$$\alpha_0 > u^{\mathrm{T}} b - 1 \tag{8}$$

$$u_i \geq 0 \quad \text{for } i = 1, \ldots, m \tag{9}$$

$$\alpha, \alpha_0 \quad \text{integer} \tag{10}$$

In the above model, the integer variables α_j and α_0 play the role of coefficients $\lfloor u^{\mathrm{T}} A_j \rfloor$ and $\lfloor u^{\mathrm{T}} b \rfloor$ in the CG cut, respectively. Hence the objective function (6) gives the amount of violation of the CG cut evaluated for $x = x^*$ that we want to maximize. Because of the sign of the objective function coefficients, the rounding conditions $\alpha_j = \lfloor u^{\mathrm{T}} A_j \rfloor$ can be imposed through upper bound conditions on variables α_j ($j = 1, \ldots, n$), as in (7), and with a lower bound condition on α_0,

as in (8). Note that this latter condition is stated as a strict inequality so as to avoid an integer $u^T b$ be rounded to $u^T b - 1$.

We now elaborate on the basic model above, in the attempt of reducing its size while improving the quality of the generated CG cuts. We observe that any variable x_j with $x_j^* = 0$ gives no contribution to the cut violation, hence there is no need to consider it explicitly in the separation model. Indeed, whenever $x_j^* = 0$ one can simply disregard x_j in the separation problem, and recompute the corresponding coefficient a posteriori, when the optimal u has been determined, as $\alpha_j := u^{\mathrm{T}} A_j$ (no time-consuming lifting operations being needed). The same holds for bounded variables x_j (if any) at their upper bound. Indeed, assuming $x_j \leq UB_j$ is part of the constraint set and $x_j^* = UB_j$, one can complement x_j before separation by replacing it with $\bar{x}_j = UB_j - x_j$ so as to have $\bar{x}_j^* = 0$—this operation of course affects the right-hand side vector b.

Moreover, we can avoid multipliers $u_i \geq 1$ that would lead to weaker cuts. Indeed, because of the integrality of (A, b) a CG cut associated with any $u_i \geq 1$ is a dominated one, as it can be obtained as the sum of the CG cut with u_i replaced by its fractional part, plus $\lfloor u_i \rfloor$ times the ith constraint $a_i^{\mathrm{T}} x \leq b_i$.

In view of the considerations above, we propose the following MIP model for CG-SEP, where an input tolerance parameter δ is used to force the coefficient fractionality to be strictly less than 1. The model contains a number of redundant constraints aimed at improving its numerical behavior as well as the quality of the generated cuts, and reads:

$$\max \sum_{j \in J(x^*)} \alpha_j x_j^* - \alpha_0 \tag{11}$$

$$f_j = u^{\mathrm{T}} A_j - \alpha_j \quad \text{for } j \in J(x^*) \tag{12}$$

$$f_0 = u^{\mathrm{T}} b - \alpha_0 \tag{13}$$

$$0 \leq f_j \leq 1 - \delta \quad \text{for } j \in J(x^*) \cup \{0\} \tag{14}$$

$$0 \leq u_i \leq 1 - \delta \quad \text{for } i = 1, \ldots, m \tag{15}$$

$$\alpha_j \text{ integer} \quad \text{for } j \in J(x^*) \cup \{0\} \tag{16}$$

where $J(x^*) := \{j \in \{1, \ldots, n\} : x_j^* > 0\}$ denotes the support of x^* (possibly after having complemented some variables and updated b accordingly). In this model we introduced explicit slack variables $f_j = u^{\mathrm{T}} A_j - \lfloor u^{\mathrm{T}} A_j \rfloor$ to capture the coefficient fractionality, and required them to be in range $[0, 1 - \delta]$ for a small $\delta > 0$. In our implementation we chose $\delta = 0.01$; this choice improves the numerical stability of our method, though it could affect the exact nature of our CG separation procedure in some pathological cases.

As already explained, the redundant constraints $u_i \leq 1 - \delta$ in (15) are intended to avoid multipliers $u_i \geq 1$ that would lead to weaker cuts. (Some unexpected practical implications of imposing the above multiplier upper bound will be discussed in Sect. 2.2.)

Model (11)–(16) can easily be adapted to deal with the case where the original model (1) contains free variables and/or equations—in this latter case, condition (15) is replaced by

$$-1 + \delta \le u_i \le 1 - \delta. \tag{17}$$

In addition, one can also allow for the presence of continuous variables. [2]

Indeed, if a variable x_j is not restricted to be integer, one can still try to derive a valid cut through model (11)–(16) by setting $\alpha_j = 0$ and imposing $u^{\mathrm{T}} A_j \ge 0$ (or $u^{\mathrm{T}} A_j = 0$ in case x_j is a free variable).

2.2 Multiplier selection

Model (11)–(16) typically has several equivalent solutions, hence we are interested in a clever way of breaking ties. Indeed, our ultimate goal is to find, for each fractional point x^* to be cut off, a "round" of Chvátal-Gomory cuts that are significantly violated and whose overall effect is as strong as possible in improving the current LP relaxation. Moreover, the separation problem should be solved as quickly as possible. In this view, we have implemented the following additional mechanisms to guide the MIP solver to provide effective cuts.

1. *Making the cut sparser* (*and stronger*). A major practical issue with CG separation is the strength of the returned cuts. As a matter of fact, several equivalent solutions of the separation problem typically exist, some of which produce very weak cuts for the ILP model (1). This is because the separation problem actually considers the face $F(x^*)$ of P_I where all the ILP constraints that are tight at x^* (including the variable bounds) are imposed as equalities. Hence for this face there exist several formulations of each cut, which are equivalent for $F(x^*)$ but not for P_I.
   To illustrate this point, consider a 2-matching problem on a graph $G = (V, E)$ and a blossom inequality $x(E(H)) + x(T) \le |H| + \lfloor |T|/2 \rfloor = 4$ for $H = \{1, 2, 3\}$ and $T = \{[1, 4], [2, 5], [3, 6]\}$, which is a CG cut with multipliers $\tilde{u}_i = 1/2$ both for the degree inequalities $x(\delta(v)) = 2$ ($v \in H$) and for the upper bound constraints $x_e \le 1$ ($e \in T$). Assume now to have a fractional point x^* with $x^*_e = 1/2$ for $e \in E(H)$, $x^*_e = 1$ for $e \in T$, and $x^*_e = 1$ for the even-cardinality cycle $C = \{[10, 11], [11, 12], \ldots, [14, 15], [15, 10]\}$. Clearly, x^* violates by 1/2 the (facet defining) blossom inequality above as well as all the CG cuts whose multiplier vector is obtained from $\tilde{u}$ by setting $\tilde{u}_i = \theta > 0$ for the multipliers on the degree inequalities on nodes $v \in \{10, 12, 14\}$, and $\tilde{u}_i = 1 - \theta$ for the multipliers on the upper bounds on all edges in C. For $\theta < 0.5$ the resulting CG cut reads $x(E(H)) + x(T) + x(C) \le |H| + \lfloor |T|/2 \rfloor + |C|$, hence it is strictly dominated by (and much weaker in practice than) the original blossom inequality, though both CG cuts correspond to optimal solutions of our separation model.

[2] The mixed-integer case is however not covered in the original Chvátal's work, and would be more appropriately addressed in the context of Gomory mixed-integer cuts.

The example above suggests that strong cuts correspond to "minimal" CG multiplier vectors with as few nonzero entries as possible.
In other words, one is interested in generating violated CG cuts associated with multiplier vectors u that are as sparse as possible—or, at least, whose components tend to satisfy a certain "minimality" condition. To this end, we modified the objective function (11) into

$$\max \left(\sum_{j \in J(x^*)} \alpha_j x_j^* - \alpha_0 \right) - \sum_{i=1}^{m} w_i u_i \tag{18}$$

by introducing the penalty term $-\sum_i w_i u_i$, where $w_i = 10^{-4}$ for all i. It is worth observing that, because of Eqs. (12)–(13), one can rewrite the violation term $\sum_{j \in J(x^*)} \alpha_j x_j^* - \alpha_0$ in (11) as $f_0 - \sum_{j \in J(x^*)} f_j x_j^* - \sum_{i=1}^{m} u_i s_i^*$, where $s^* = b - Ax^* \geq 0$. Therefore, all variables u_i associated with slack constraints with $s_i^* > 0$ are already penalized (implicitly) in the objective function, so the explicit penalty term w_i is only relevant for tight constraints, and can be omitted when $s_i^* > 0$.

2. *Enhancing cut diversification*. Preliminary computational experiments have shown the presence of the lower/upper bounds (15) or (17) on the multipliers u_i sometimes weaken the overall performance of our method. This is true, in particular, for the constraints stated in equality form—removing the conditions $-1 < u_i < 1$ often produces better results. This outcome is somehow surprising since, as already discussed, removing the multiplier bounds tends to produce much denser (and potentially weaker) cuts. A possible explanation of this behavior is that the removal of the redundant lower/upper bounds on the CG multiplier is beneficial for the MIP heuristics we use (`ILOG-Cplex` 9.1). Even more importantly, it seems that leaving more freedom in the choice of the multipliers produces a useful "diversification effect" in breaking ties between equivalent solutions of the separation model (11)–(16), hence each call of the MIP separation tends to return cuts that are less correlated with those generated in the previous calls.
   Therefore, in our final implementation we allow for the maximum degree of freedom, and remove all the lower/upper bound conditions (15) or (17) on the CG multipliers (we only keep the nonnegativity conditions associated with constraints in less-than-or-equal-to form, as they are required for the correctness of the method). The underlying idea is that, for slack cuts with $s_i^* > 0$, the objective function already tends to reduce the absolute value of the associated multiplier, hence the lower/upper bounds on the multiplier are somehow redundant. For tight constraints, instead, imposing the bound conditions is immaterial as far as the separation of the *current* point x^* is concerned, but hurts the MIP heuristics as it reduces their degree of freedom. A drawback of this choice is however that (in spite of the penalty terms in the objective function) the returned cuts may involve multipliers u_i with

$|u_i| > 1$, i.e., the cuts may be unnecessarily dense (and quite weak, in case of less-than-or-equal-to constraints). We therefore implemented a simple cut post-processing procedure where each CG multiplier returned by the MIP solver is replaced by its fractional part (when this is mathematically correct) before deriving the actual CG cut. As shown in the computational tests, this mechanism turns out to be quite effective, in particular, for the problems involving a large number of equality (i.e., always tight) constraints.

2.3 Implementation in a pure cutting-plane framework

We have embedded our CG cut separator into a pure cutting-plane framework where we keep generating violated CG cuts of rank 1 (with respect to the original formulation of the ILP model at hand), until we stop either because no such violated cut exists (in which case we have optimized over the first closure), or because an overall time-limit condition is met. Moreover, we implemented the following simple mechanisms:

- When the LP relaxation of the original ILP model (1) is solved, we take all the violated Gomory fractional cuts that can be read from the current tableau, and skip CG separation; in principle this mechanism could also be applied in the subsequent iterations, provided that we skip the Gomory cuts having a non-zero CG multiplier for a constraint that was not in the original formulation, so as to ensure that all generated cuts are of rank 1.
- The MIP solver for CG separation is invoked with an initial lower bound of 0.01, meaning that we are only interested in CG cuts violated by more than 0.01.
- Each time the MIP solver for CG separation updates its incumbent solution, we store the corresponding CG cut in a pool, with the aim of adding it to the current ILP formulation right after the end of the separation phase. Because of the penalty term that appears in the MIP objective function (11), however, we found that it is useful to skip all the CG cuts with the same violation $\sum_{j \in J(x^*)} \alpha_j x_j^* - \alpha_0$, except the one with the sparsest support.
- The MIP execution for CG separation is stopped if either the optimal solution has been found, or a prefixed number τ of branching nodes has been explored after the last update of the incumbent solution, where $\tau = 1,000$ if the violation of the incumbent is less than 0.2, and $\tau = 100$ if the violation is greater or equal to 0.2.

3 Why?

We next try to give concrete (i.e., supported by computational results) answers to some important questions related to the practical behavior of our MIP separator and to the effectiveness of rank-1 CG cuts, when applied to a wide range of ILP instances—both with and without a specific combinatorial structure.

Our implementation of the cutting-plane method uses the commercial software `ILOG-Cplex` 9.1 as the LP solver, whereas the separation problem is solved through `ILOG-Cplex` 9.1 MIP solver with "mip emphasis 4" parameter; see [17]. All computing times refer to a Pentium M 1.6 GHz notebook with 512 MByte RAM.

3.1 Code tuning on matching problems

In order to investigate the performance of our separation tool and tune its main parameters (and solution mechanisms) in a "familiar" setting, we addressed first the Edmonds–Johnson [12] *matching problem*, defined as

$$\min\{c^{\mathrm{T}}x:\ Ax=b,\ 0\le x\le q,\ x \text{ integer}\} \tag{19}$$

where A, b, q are integer arrays of appropriate dimensions, and $\sum_{i=1}^{m}|a_{ij}|\le 2$ holds for all $j=1,\ldots,n$. A celebrated result of Edmonds [11,12] says that P_1 is an integer polyhedron, i.e., P_1 and P_I coincide for matching problems. Moreover, undominated Chvátal cuts (called *blossom* inequalities) only arise for $u\in\{0,1/2\}^m$, and correspond to setting $u_i=1/2$ for (a) the equations $a_i^{\mathrm{T}}x=b_i$ associated with a certain set H (say) and (b) the upper-bound constraints $x_j\le q_j$ associated with a possibly-empty set $T\subseteq\{1,\ldots,n\}$, such that $\sum_{i\in H}b_i+\sum_{j\in T}q_j$ is an odd number. Padberg and Rao [20] have shown that blossom inequalities can be separated in polynomial time by an ad-hoc procedure looking for a minimum-weight odd cut in a certain graph; an alternative algorithm with an improved running time has been proposed recently in Letchford, Reinelt and Theis [18].

A relevant case of matching problems arises when A is the node-edge incidence matrix of a (multi-)graph $G=(V,E)$. Given edge costs c_e $(e\in E)$ and node values $b_v\ge 0$ $(v\in V)$, we call for a minimum-cost partial graph $G'=(V,M)$ of G where each node v has degree b_v. The *b-matching problem* can then be formulated as

$$\min\left\{c^{\mathrm{T}}x:\ \sum_{e\in\delta(v)}x_e=b_v \text{ for } v\in V,\ 0\le x\le 1,\ x \text{ integer}\right\}. \tag{20}$$

Also belonging to the family of matching problems is the *T-join problem*, where each node v has an associated parity $b_v\in\{0,1\}$ and one looks for a minimum-cost partial graph $G'=(V,M)$ of G where each node v has an odd degree if and only if $b_v=1$. This problem can be formulated as the b-matching problem above, with the degree equations replaced by $\sum_{e\in\delta(v)}x_e-2z_v=b_v$, z_v integer, for all $v\in V$.

Table 1 reports computational results on a set of 2-matching problems taken from TSPLIB. For each instance, the table gives the problem name (ID), the initial LP relaxation value (initial LB), the value of the optimal integer solution

Table 1 2-Matching problems

ID	Initial LB	*Optimum*	# iter.s	# cuts	*CPUtime*
eil101	619.0	623.0	17	34	0.95
gr120	6,662.5	6,694.0	38	49	6.14
pr124	50,164.0	51,477.0	83	141	13.96
gr137	66,643.5	67,009.0	15	37	1.26
pr144	32,776.0	33,652.0	35	71	3.86
ch150	6,281.0	6,337.0	76	125	44.41
rat195	2,272.5	2,297.0	125	231	160.74
kroA200	27,053.0	27,426.0	26	81	5.56
kroB200	27,347.0	27,768.0	171	332	215.37
ts225	115,605.0	121,261.0	613	1,401	2,914.36
pr226	55,247.5	57,177.0	230	319	166.83
gr229	127,411.0	128,353.0	96	176	87.37
gil262	2,222.5	2,248.0	150	304	195.89
a280	2,534.0	2,550.0	89	129	34.89
lin318	38,963.5	39,266.0	321	589	4,046.93

(Optimum), the number of calls to our separation procedure (# iter.s), the total number of separated CG cuts (# cuts), and the total CPU time in seconds.

In all cases, our method was able to compute the optimal solution over the first Chvátal closure, and produced an integer solution—as expected.

Table 2 shows that the multiplier-selection rules discussed in Sect. 2.2 are indeed effective in reducing the overall computing time. In the table, we compare the results of our best algorithm (column *best version*) with two alternative implementations where either the mechanism for making the cuts sparser (column *no sparsification*) or more diversified (column *no diversification*) has been disabled. For each run, a limit of 100 separation calls has been imposed. The table contains the same information as in Table 1, plus the percentage of integrality gap closed (*% gap closed*) computed as customary as

$$100 - 100(\text{opt\_value}(P_I) - \text{opt\_value}(P_1)/\text{opt\_value}(P_I) - \text{opt\_value}(P)).$$

Table 2 clearly shows the performance deterioration resulting, in particular, from the removal of the diversification mechanism.

It should be observed that some of these instances can be solved in a much shorter computing time by just applying `ILOG-Cplex` 9.1 MIP solver to the rank-0 model (20). However, for some hard instances our method also has a practical interest if used in a so-called *cut-and-branch* approach where we abort our cut generation after the addition a certain number of CG cuts, and then switch to a commercial MIP solver for concluding the optimization. This is illustrated in Table 3, where some large 2-matching instances are solved through `ILOG-Cplex` MIP solver with and without the addition of an initial pool of CG cuts generated by 100 calls to our separation procedure and using a purging routine to keep the size of the LP rather small.

Table 2 2-Matching problems: impact of the multiplier selection rules

ID	Best version				No sparsification				No diversification			
	# iter.s	# cuts	CPU time	% Gap closed	# iter.s	# cuts	CPU time	% Gap closed	# iter.s	# cuts	CPU time	% Gap closed
eil101	17	34	1.03	100.0	17	34	1.06	100.0	100	217	40.60	92.6
gr120	38	49	6.45	100.0	42	62	12.88	100.0	100	277	104.50	83.4
pr124	83	141	14.57	100.0	93	134	22.52	100.0	100	210	41.40	25.6
gr137	15	37	1.37	100.0	15	37	1.42	100.0	100	221	62.00	77.2
pr144	35	71	4.29	100.0	38	79	8.05	100.0	100	231	46.50	65.5
ch150	76	125	45.91	100.0	81	154	75.26	100.0	100	276	111.90	57.7
rat195	100	171	79.20	97.4	100	194	104.40	93.3	100	292	167.90	34.8
kroA200	26	81	6.21	100.0	26	81	6.34	100.0	100	250	87.50	80.3
kroB200	100	175	39.60	82.2	100	171	39.20	77.0	100	229	87.70	30.9
ts225	100	172	51.70	84.1	100	179	54.00	88.8	100	245	102.00	37.6
pr226	100	170	33.90	96.1	100	152	31.40	80.8	100	195	73.50	31.3
gr229	96	176	92.60	100.0	100	220	145.70	97.4	100	249	156.40	46.3
gil262	100	194	59.20	97.0	100	202	103.10	96.7	100	247	112.30	48.5
a280	89	129	39.47	100.0	70	100	33.84	100.0	100	176	102.00	50.0
lin318	100	169	73.30	68.0	100	183	93.90	69.4	100	273	210.20	26.6

Table 3 2-Matching problems: cut-and-branch approach

ID	ILOG-Cplex			Cut-and-branch				
	% Gap closed	Nodes	Time	# cuts	% Gap closed	Separation time	Nodes	Total time
kroB200	100.0	330,913	2,748.24	179	77.1	39.20	348	60.67
ts225	47.1	230,115	1h	193	85.9	46.10	1,018	75.39
pr226	55.0	288,901	1h	168	91.3	31.60	51	35.20
gr229	100.0	15,005	180.79	182	96.9	54.70	7	60.94
gil262	100.0	117,506	2,094.77	172	97.4	54.50	1	58.39
lin318	53.3	117,100	1h	182	69.5	62.60	24,734	814.74

3.2 How tight is the first closure for MIPLIB instances?

In this section we aim at evaluating the best-possible effect of rank-1 CG cuts in closing the integrality gap for MIPLIB instances, by actually optimizing the problem objective function over P_1. In this respect, CG cuts indeed proved very effective in most cases, as they are able to close a large percentage of the integrality gap—a piece of information that was not available before the present study.

Tables 4 and 5 report the outcome of our experiments on a test-bed made by all the pure-integer problems from MIPLIB 3.0 and 2003, respectively. For the instances in the MIPLIB 3.0 (resp., MIPLIB 2003), the cases where our cutting plane procedure exceeded the overall time limit of 3 (resp., 12) hours are given in the bottom part of Table 4 (resp., Table 5). In this situation, the percentage of integrality gap closed (% gap closed) is underestimated as we only have a

Table 4 General IPs of the MIPLIB 3.0

ID	Optimum	# iter.s	# cuts	% Gap closed	Time
air03	340,160.00	1	35	100.0	0.65
lseu	1,120.00	95	268	93.3	174.90
mod008	307.00	29	118	100.0	12.25
mod010	6,548.00	5	40	100.0	0.74
p0033	3,089.00	42	115	85.3	16.20
stein27	18.00	142	392	0.0	521.00
gt2	21,166.00	31,733	64,436	$\geq$ 91.0	3 h
enigma	0.00	13,234	53,635	$--$	3 h
l152lav	4,722.00	793	2,249	$\geq$ 59.6	3 h
misc03	3,360.00	428	1,154	$\geq$ 51.0	3 h
mitre	115,155.00	320	3,010	$\geq$ 16.2	3 h
p0201	7,615.00	911	2,568	$\geq$ 60.6	3 h
p0282	258,411.00	641	1,920	$\geq$ 99.9	3 h
p0548	8,691.00	2,327	5,493	$\geq$ 62.4	3 h
stein45	30.00	14,463	31,643	$\geq$ 0.0	3 h

For instance `enigma` there is no gap between the value of the initial (fractional) LP solution and the optimal value. Instance `misc03` has been preprocessed to get rid of a single continuous variable

lower bound on the optimal value opt_value(P_1) over the first closure. Finally, for the instances in Table 5 denoted with '**', the optimal solution is unknown and column 'Optimum' gives the best known solution.[3]

The impact of the mechanisms for making the cuts sparser and more diversified, as described in Sect. 2.2, is in the MIPLIB context less impressive. In particular, the diversification effect guaranteed by removing the multiplier lower/upper bounds appears crucial just for a few specific instances. For example, if one does not remove the multiplier bounds when solving problem `mitre`, the first 100 separations are completely ineffective in improving the lower bound value of 114,759.48 obtained after the addition of the Gomory cuts read from the initial tableau—though 1,610 new cuts are added. As a comparison, the version without multiplier bounds brings the lower bound to 114,775.35 after the same number of iterations, i.e., it closes 8.4% of the gap (more than a half of the final gap closed within the time limit). In general, however, the overall impact is not as strong as the one shown in Table 2 for the 2-matching case. One possible explanation is that the models in the MIPLIB are more heterogeneous (in terms of constraints) and involve fewer equations with respect to the 2-matching ones, hence our diversification mechanism becomes less important. The use of the penalty terms for making the cuts sparser appears instead quite crucial for the MIPLIB instances too. For example, for problem `fiber` our best implementation closes 91.0% of the initial gap after 100 iterations, while the version without the penalty terms closes only 33.9% of the gap in the same number of iterations. This trend is consistently confirmed on the other instances.

[3] Thanks are due to Tobias Achterberg for providing some of these solution values.

Table 5 General IPs of the MIPLIB 2003. For instance `disctom` there is no gap between the value of the (fractional) initial LP solution and the optimal value

ID	Optimum		# iter.s	# cuts	% Gap closed	Time
manna81	−13,164.00		1	670	100.0	1.48
nw04	16,862.00		127	492	100.0	509.27
air04	56,137.00		100	713	≥ 30.4	12 h
air05	26,374.00		450	2,019	≥ 35.3	12 h
cap6000	−2,451,377.00		234	302	≥ 22.5	12 h
disctom	−5,000.00		682	4,585	–	12 h
ds	283.44	**	108	1,209	≥ 0.3	12 h
fast0507	174.00		39	292	≥ 5.3	12 h
fiber	405,935.18		623	2,361	≥ 99.0	12 h
harp2	−73,899,798.00		2,458	3,251	≥ 49.5	12 h
misc07	2,810.00		1,145	3,033	≥ 19.0	12 h
mzzv11	−21,718.00		67	2,501	≥ 12.1	12 h
mzzv42z	−20,540.00		46	1,761	≥ 13.3	12 h
p2756	3,124.00		2,357	5,541	≥ 42.6	12 h
protfold	−30.00	**	492	2,339	≥ 2.7	12 h
seymour	423.00		294	3,790	≥ 33.0	12 h
sp97ar	664,565,103.76	**	595	1,465	≥ 30.4	12 h
t1717	193,221.00	**	137	1,317	≥ 0.3	12 h

Instances `mzzv11` and `mzzv42z` have been scaled to get rid of negative bounds on some variables. Instances `fiber` and `misc07` have been preprocessed to get rid of a single continuous variable. Instance `stp3d` is not reported in the table since no feasible solution is known for it: our algorithm could improve the initial bound only slightly (from 481.88 to 481.93) although 2,926 cuts have been separated

An interesting experiment compares the quality of the CG cuts readable from the LP tableau by using the classical Gomory procedure, with that of the rank-1 CG cuts generated through our MIP separation. Table 6 gives the outcome of that experiments by reporting the percentage gap closed by 1, 5, and 10 rounds of CG cuts (possibly of rank greater than 1) read from the tableau, with the percentage gap closed by our rank-1 CG cuts—as reported already in Tables 4 and 5.

Table 6 confirms the high quality of the CG cuts of rank 1 obtained by our procedure. Indeed, the percentage gap closed by rank-1 cuts generally outperforms that resulting from the addition of 10 rounds of CG cuts taken from the tableau, with the only exception of 4 instances (`mitre`, `mzzv11`, `mzzv42z` and `protfold`).

Finally, in order to evaluate the viability of using our cut generator in a cut-and-branch approach akin to the one we described for 2-matching problems, we made the following experiment on the very hard ILP instance *harp2*. After having generated 211 rank-1 CG cuts (53 of which are tight at the end of the cutting-plane phase) in 100 rounds of separation, we switched to `ILOG-Cplex` MIP solver and obtained the optimal solution within 1,500 CPU seconds and 400K nodes (including both cut generation and branching), while `ILOG-Cplex`

Table 6 CG cuts read from the tableau versus rank-1 CG cuts

	% Gap closed			
	CG cuts from the tableau			Rank-1
ID	1 Round	5 Rounds	10 Rounds	CG cuts
air03	100.0	100.0	100.0	100.0
enigma	−−	−−	−−	−−
gt2	0.1	32.2	33.9	91.0
l152lav	2.5	7.8	≥ 29.2	≥ 59.6
lseu	6.9	11.5	11.8	93.3
misc03	6.4	19.0	19.0	≥ 51.0
mitre	4.6	40.2	77.9	16.2
mod008	0.7	6.6	10.2	100.0
mod010	84.9	100.0	100.0	100.0
p0033	4.3	68.4	77.9	85.3
p0201	3.4	13.5	31.0	≥ 60.6
p0282	0.3	0.6	≥ 0.6	≥ 99.9
p0548	3.3	11.9	17.0	62.4
stein27	0.0	0.0	0.0	0.0
stein45	0.0	0.0	0.0	≥ 0.0
air04	4.0	≥ 11.9	≥ 12.5	≥ 30.4
air05	2.7	≥ 4.5	≥ 4.6	≥ 35.3
cap6000	0.9	3.4	4.9	≥ 22.5
disctom	−−	−−	−−	−−
ds	0.0	≥ 0.0	≥ 0.0	≥ 0.3
fast0507	1.5	≥ 2.3	≥ 2.4	≥ 5.3
fiber	10.0	40.7	≥ 51.3	≥ 99.0
harp2	1.3	5.5	≥ 6.4	≥ 49.5
manna81	100.0	100.0	100.0	100.0
misc07	0.7	0.7	≥ 0.7	≥ 19.0
mzzv11	6.3	≥ 13.4	≥ 20.4	≥ 12.1
mzzv42z	8.0	≥ 18.4	≥ 21.0	≥ 13.3
nw04	56.8	100.0	100.0	100.0
p2756	0.2	0.6	≥ 42.9	≥ 42.6
protfold	2.7	≥ 4.5	≥ 5.3	≥ 2.7
seymour	7.1	≥ 15.6	≥ 15.8	≥ 33.0
sp97ar	6.1	≥ 8.5	≥ 9.1	≥ 30.4
t1717	0.0	0.0	0.0	≥ 0.3

For the cases where the % gap closed by 5 or 10 rounds of CG cuts from the tableau is given as a '≥' inequality (due to saturation of the 4-GByte main memory of our computer), only 500 or 5,000 most violated cuts (depending on the instance size) have been added

alone required more than 15,000 CPU seconds and 7M nodes to solve the original instance.

3.3 Beyond the first closure?

The previous experiments show that optimizing over the first closure often gives a very tight approximation of the integer optimal value, though it may require a large computing time. Besides the obvious research topic of designing more specific separation procedures for CG cuts, we addressed the possibility of using

our cutting plane method as a pre-processing tool, to be used to strengthen the user's formulation by possibly exploiting cuts of Chvátal rank larger than 1. This idea was evaluated by comparing two different cut preprocessors, namely:

- *cpx*. Apply `ILOG-Cplex` 9.1 (with mip emphasis "move best bound") on the current ILP model, save the final root-node model (including the generated cuts) in a file, and repeat on the new model until a total time limit is exceeded.
- *cpx-cg*. Apply `ILOG-Cplex` 9.1 (with mip emphasis "move best bound") on the current ILP model, followed by 600 seconds of our CG separation procedure; then save in file the ILP model with all the cuts that are active in the last LP solution,[4] and repeat on the new model until a total time limit is exceeded.

Figure 1 compares the behavior of the two methods above on a very hard instance from MIPLIB 2003, namely *timtab1* (viewed here as a pure ILP problem, as it is correct in this case). The horizontal axis counts the number of calls to the separation procedures (not the computing time nor the number of generated cuts), hence comparing the graph slopes can be misleading since every separation call produced about 10–50 cuts for *cpx*, but just 1–5 cuts for *cpx-cg*. Even taking these considerations into account, however, it is clear from the figure that method *cpx* exhibits an early tailing-off phenomenon, while the growth for *cpx-cg* is more regular and produces a much better final lower bound.

Remarkably, our cut preprocessor allowed us to find, for the first time [1], a provable optimal solution of value 51,200.00 for the very hard instance *nsrand-ipx* (in order to get a pure ILP, the objective function $x1$ was replaced by its expression given by the first constraint in the model.) To this end, we applied `ILOG-Cplex` 9.1 MIP solver to the tightened formulation returned by *cpx-cg* after 4,800 s—this brought the initial LP bound from 49,667.89 to 50,665.71.

3.4 Can we discover new classes of strong inequalities?

The capability of separating over the CG cut class also has an interesting application in the off-line study of the facial structure of a specific problem. This is in the spirit of the widely-used approach of producing (through appropriate software such as PORTA [8]) an explicit polyhedral description of very small instances of the problem at hand, but has the advantage of being applicable to instances of much larger size.

To illustrate a possible application to the *Asymmetric Travelling Salesman Problem* (ATSP), we made the following experiment. We set up a partial ATSP formulation including out- and in-degree equations, plus the subtour elimination constraints on 2-node sets, i.e., we addressed the NP-hard *Asymmetric Assignment Problem* (AAP) relaxation studied by Balas [2]. We then applied our separation procedure to a specific ATSP instance from TSPLIB, namely

[4] As we no longer insist in staying within the first Chvátal closure, CG cuts could be improved by deriving the stronger Gomory mixed-integer cut from the surrogate equations $u^{\mathrm{T}}A+s = u^{\mathrm{T}}b$, $s \geq 0$.

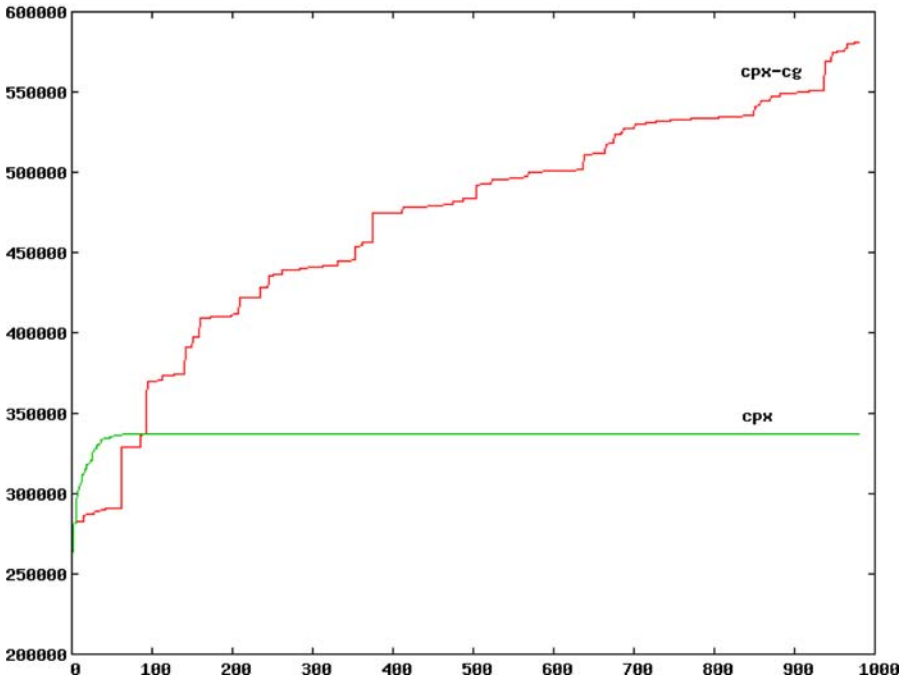

Fig. 1 Lower bounds provided by *cpx* and *cpx-cg* after each call of the separation procedures, for the hard MIPLIB instance *timtab1*

the 48-node instance *ry48p*, and stored the CG cuts we found along with the associated CG multipliers.[5] One of the returned CG cuts had a violation of 0.75, and was associated with the following non-zero CG multipliers (reported in parenthesis): out-degree equation for nodes 21 (c0.5), 35 (0.5), 47 (−0.25), 2 (0.5), 26 (0.5), 12 (−0.25), 20 (−0.25), and 48 (−0.5); in-degree equations for nodes 5 (−0.5), 29 (−0.5), 35 (−0.5), 47 (0.25), 4 (−0.5), 42 (−0.5), 32 (0.5), 33 (0.25), and 20 (0.25); subtour elimination constraint on the 2-node sets $\{5, 29\}$ (0.5), $\{21, 47\}$ (0.25), and $\{35, 45\}$ (0.5). To simplify our analysis, we normalized the corresponding cut by adding 1 to all the negative CG multipliers, thus producing an equivalent CG cut in the so-called *h-canonical form* [3], i.e., with non-negative and relative prime coefficients, whose support leaves at least one node h isolated. This form immediately showed the presence of *clones* [3], that for the purpose of our analysis could be removed from the digraph. After renaming the nodes, we were left with an 8-node digraph and with the following CG multipliers: out-degree equation for nodes 1 (0.75), 3 (0.75), 4 (0.5), 5 (0.75), and 6 (0.5); in-degree equations for nodes 1 (0.25), 2 (0.5), 5 (0.25), 6 (0.5), and 7 (0.25); subtour elimination constraint on the 2-node sets $\{4, 5\}$ (0.25) and $\{6, 7\}$ (0.5). The associated CG cut is $\alpha^{\mathrm{T}} x \leq \alpha_0$, where

[5] For CG cuts in the form returned by our procedure, these multipliers could easily be recomputed a posteriori, by solving an LP.

$$\alpha = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}; \quad \alpha_0 = 5 \tag{21}$$

and it is easily shown (by computational methods) to be facet-defining for ATSP (and hence for AAP). Using *clique lifting* [3] we can then obtain a large class of ATSP facets, that to the best of our knowledge is new.

Acknowledgments Part of this research has been carried out when the second author was Herman Goldstine Fellow of the IBM T.J. Watson Research Center, whose support is strongly acknowledged. Thanks are due to Jon Lee and François Margot for useful discussions, and to two anonymous referees for helpful comments.

References

1. Achterberg, T., Koch, T., Martin, A.: The mixed integer programming library: MIPLIB 2003, http://www.miplib.zib.de (2003)
2. Balas, E.: The asymmetric assignment problem and some new facets of the traveling salesman polytope on a directed graph. SIAM J. Discrete Math. **2**, 425–451 (1989)
3. Balas, E., Fischetti, M.: A lifting procedure for the Asymmetric Traveling Salesman Polytope and a large new class of facets. Math. Program. **58**, 325–352 (1993)
4. Balas, E., Saxena, A.: Optimizing over the split closure, Technical Report 2006-E5, Tepper School of Business, CMU (2005)
5. Bonami, P., Cornuejols, G., Dash, S., Fischetti, M., Lodi, A.: Projected Chvatal-Gomory cuts for mixed integer linear programs. Technical Report 2006-E4, Tepper School of Business, CMU, to appear Math. Program. (in press)
6. Bixby, R.E., Ceria, S., McZeal, C.M., Savelsbergh, M.W.P.: MIPLIB 3.0, http://www.caam.rice.edu/∼bixby/miplib/miplib.html
7. Caprara, A., Letchford, A.N.: On the separation of split cuts and related inequalities. Math. Program. **94**, 279–294 (2003)
8. Christof, T., Löbel, A.: PORTA - POlyhedron representation transformation algorithm, http://www.zib.de/Optimization/Software/Porta/
9. Chvátal, V.: Edmonds polytopes and a hierarchy of combinatorial problems.. Discrete Math. **4**, 305–337 (1973)
10. Dash, S., Günlük, O., Lodi, A.: On the MIR closure of polyhedra. IBM, T.J. Watson Research, Working paper, (2005)
11. Edmonds, J.: Maximum matching and a polyhedron with {0,1}-vertices. J. Res. Nat. Bur. Stand. B **69**, 125–130 (1965)
12. Edmonds, J., Johnson, H.L.: Matching: a well-solved class of integer linear programs. In: Guy, R.K., et al. (eds.) Combinatorial Structures and their Applications, pp. 89–92. Gordon and Breach, New York (1970)
13. Eisenbrand, F.: On the membership problem for the elementary closure of a polyhedron. Combinatorica **19**, 297–300 (1999)
14. Fischetti, M., Lodi, A.: Optimizing over the first Chvátal closure. In: Jünger, M., Kaibel, V. (eds.) Integer programming and combinatorial optimization—IPCO 2005, LNCS 3509, pp. 12–22. Springer, Berlin Heidelberg New York (2005)

15. Gomory, R.E.: Outline of an algorithm for integer solutions to linear programs. Bull. AMS **64**, 275–278 (1958)
16. Gomory, R.E.: An algorithm for integer solutions to linear programs. In: Graves, R.L., Wolfe, P. (eds.) Recent Advances in Mathematical Programming, pp. 275. McGraw-Hill, New York (1963)
17. ILOG Cplex 9.1: User's manual and reference manual, ILOG, S.A. http://www.ilog.com/(2005)
18. Letchford, A.N., Reinelt, G., Theis, D.O.: A faster exact separation algorithm for blossom inequalities. In: Bienstock, D., Nemhauser, G. (eds.) Integer programming and combinatorial optimization—IPCO 2004, LNCS 3064, pp. 196–205. Springer, Berlin Heidelberg New York (2004)
19. Nemhauser, G.L., Wolsey, L.A.: Integer and Combinatorial Optimization. Wiley, New York (1988)
20. Padberg, M.W., Rao, M.R.: Odd minimum cut-sets and *b*-matchings. Math. Oper. Res. **7**, 67–80 (1982)