On the generation of cutting planes which maximize the bound improvement

Stefano Coniglio, Martin Tieves
Lehrstuhl II für Mathematik, RWTH Aachen University
{coniglio,tieves}@math2.rwth-aachen.de

September 23, 2014

Note: this paper, in its current incarnation, is a largely updated version of [Con13], but still not up to date to the presentation given by the first author at AIRO 2014.

Abstract

We propose the bound-optimal cutting plane method. It is a new paradigm for cutting plane generation in Mixed Integer Programming allowing for the simultaneous generation of k cuts which, when added to the current Linear Programming relaxation, yield the largest bound improvement. By Linear Programming duality arguments and standard linearization techniques we show that, for a large family of cutting planes, the cut generating problem of our cutting plane method can be formulated as a Mixed 0-1 Integer Program.

We highlight the potential of our technique by presenting computational experiments on the generation of bound-optimal stable set inequalities for the fractional clique number problem. We compare our method to the standard generation of maximally violated cuts, to the generation of cutting planes corresponding to a steepest-edge pivot in the dual, and to coordinated cutting plane generation. With respect to the standard algorithm, the bound-optimal cutting plane method allows for a substantial reduction in the number of cutting plane iterations and/or cuts needed to achieve either a given bound or an optimal solution. This reduction is still substantial also in comparison to the other techniques.

1 Introduction

Consider an Integer (Linear) Program (IP) over n variables, namely: $P_{IP} := \max\{cx : Ax \leq b, x \in \mathbb{Z}^n_+\}$. For all the developments of this paper, the extension to the mixed-integer case is straightforward. Let $P_{LP} := \max\{cx : Ax \leq b, x \in \mathbb{Q}^n_+\}$ be the Linear Programming (LP) relaxation of P_{IP}. Let $P_{\mathcal{C}}$ be the relaxation obtained when adding to P_{LP} all the valid inequalities $\alpha x \leq \alpha_0$ belonging to a family $\mathcal{C}$. Throughout the paper, we will adopt the notation $(\alpha, \alpha_0) \in \mathcal{C}$. Assume that, rather than solving P to optimality, we focus on solving $P_\mathcal{C}$, where $P_{IP} \subseteq P_\mathcal{C} \subseteq P_{LP}$.

We tackle $P_\mathcal{C}$ via a cutting plane algorithm where the inequalities in $\mathcal{C}$ are introduced one at a time by solving a cut generating problem. As the solutions to $P_\mathcal{C}$ are not integer constrained, we never resort to branching, thus considering a *pure* cutting plane method. Typically, the cut generating problem amounts to a *separation problem* where, given an optimal solution x^* to the current relaxation (a superset of $P_\mathcal{C}$ containing the inequalities from $\mathcal{C}$ added so far), we look for an inequality in $\mathcal{C}$ which is violated (that is, a *cut*), if any. Even though the introduction of any violated inequality is sufficient to guarantee the convergence of a cutting plane method, it is usual to solve a separation problem where we generate one which is *maximally violated*, that is, which maximizes the linear function $\alpha x^* - \alpha_0$. We will refer to such separation problem (and to the corresponding cutting plane method) as to the *standard* one. Note that the current practice for cut generation in state-of-the-art solvers, like CPLEX, Gurobi, or SCIP, is somewhat more involved. More families of cuts are usually considered, separation is typically carried out heuristically, and more cuts are introduced into the LP relaxation at the same time. Such cuts are usually only a fraction of those that have been generated, due to being filtered by some cut selection procedure. In this work, so to better understand the impact of the cutting plane algorithm that is adopted, we opt for a cleaner setting which is not affected by the influence of branching or other components of a typical integer programming solver.

This paper belongs to a larger stream of work, started in [ZFB11] and continued in [ACG12], where alternative paradigms for cutting plane generation are sought. The aim is of exploring alternatives to the maximization of the cut violation to obtain an improvement in, at least in the first place, the number of iterations needed to achieve a certain bound.

In this work, we do not directly look for a method which reduces the computing time needed to solve a problem. Rather, we propose a cutting plane method which, in practice, requires substantially less iterations and cuts than the standard one to achieve a given bound. Our method, as implemented and applied in the paper, is not time efficient. Even if heuristic procedures could be adopted to produce a faster algorithm, we do not resort to them here, so to better understand the full potential of the method. With this work, we aim at showing that there is vast room for improvement over cutting plane algorithms based on cut violation, hopefully motivating future developments of efficient techniques which are based on our proposal.

1.1 Motivation

To convince the reader that there are better options than looking for maximally violated cuts, we exploit a simple relationship: the correspondence between, in the primal simplex method, pivoting on a nonbasic column and, in a cutting plane method, generating a cutting plane among those that are in $\mathcal{C}$. Indeed, one can clearly look at the cutting plane algorithm as to a method which solves an LP with a (possibly exponentially) large number of inequalities which are introduced one at a time into the formulation. In this sense, the method precisely amounts to solving $P_\mathcal{C}$ with the dual simplex method where the introduction of a new violated inequality in the former amounts to pivoting on the corresponding (violated) row in the latter. When adopting a dual point of view, the cutting plane method corresponds, equivalently, to solving the dual of $P_{\mathcal{C}}$. The latter is an LP with a (possibly exponentially) large number of columns, which is solved by introducing them one at a time into the formulation (as in a column generation method). What is more, as we will recall in Section 2, adding a maximally violated cutting plane among those in $\mathcal{C}$ and reoptimizing amounts to pivoting, in the primal, on a row with the most negative slack or, in the dual, on a column with the most negative reduced cost. Most importantly, in the simplex method such pivoting rule is usually considered a poor one, as reported, among other sources, in [Har73]:

> At the very inception of linear programming, Dantzig realized that the criterion of most negative reduced cost for selecting a new basic variable, chosen for computational ease, was not necessarily the best. He preferred the "greatest change" criterion [...].

The first part of the quotation is backed up by computational experience, which suggests that pivoting on the column with the most negative reduced cost, which in the cutting plane case amounts to introducing the most violated inequality, is in practice as poor a choice as pivoting on a random column with a negative reduced cost, see [Bix09]. Curiously, such pivoting rule often goes by the name of *Dantzig's rule*. The second part of the quotation refers to pivoting on a column which yields the largest improvement in the objective function. In this sense, the "greatest change" criterion yields a greedy simplex method where, iteration by iteration, the best improving solution within a pivoting operation is sought. To the best of our knowledge, efficient ways to find such an improving pivot are not known. Nevertheless, the underlying idea has given rise to alternative pivoting rules which, although only providing an approximation of the actual objective function improvement, are very effective in practice. As we will remark in Section 2, steepest-edge pricing, see [GR77] and [FG92], is one such rule.

In the case of linear programming, when the LP is fully available *a priori* in an explicit form of manageable size, the pivot yielding the "greatest change" could be found by tentatively pivoting on every nonbasic variable and then backtracking to choose the most improving one. Besides the obvious inefficiency of such an implementation, it cannot be applied in a cutting plane setting where the cuts are not available *a priori*.

1.2 Contribution and outline of the paper

Inspired by Dantzig's "greatest change" criterion, in this work we propose a way to generate, at the same time, a set of k cutting planes which simultaneously yield the largest bound improvement. We refer to the cutting plane method arising from this paradigm as to the *bound-optimal cutting plane method*.

The paper is organized as follows. In Section 2, we recall some preliminary notions concerning pivoting in the simplex method and formalize the relationship between reduced-cost pivoting and the generation of maximally violated cuts. In this context, we review some alternative cutting plane paradigms from the literature, among which cut coordination, recently proposed in [ACG12]. We also discuss how to apply steepest-edge pricing in the context of a cutting plane method. In Section 3, we formally introduce the bound-optimal cutting plane method. We show how to generate bound-optimal cuts with a Quadratically Constrained Mixed-Integer Program in the general case and, for cuts with binary coefficients, via a Mixed-Integer Linear Program (MILP). In Section 4, we adapt our method to the Fractional Clique Number problem for which, in Section 5, computational results are reported. We conclude with Section 6, summarizing our contribution and and highlighting some future developments. A preliminary, partial version of this work appeared in [Con13].

2 On the relationship between cutting plane generation and pivoting, and alternatives to cut-violation

In this section, we formalize the connection between cutting plane generation and pivoting, first analyzing the case of reduced-cost pivoting. We then consider that of steepest-edge pricing, illustrating its adaptation to the context of cutting plane generation. Lastly, we review other alternatives to maximizing the cut violation in the separation problem, as proposed in the recent cutting plane literature.

2.1 Maximally violated cuts and reduced-cost pivoting

Consider an LP in standard form, namely: $P := \min\{cx : Ax = b, x \geq 0\}$, where $A \in \mathbb{Q}^{m \times n}$ and $m \leq n$. Its dual reads $D := \max\{by : yA \leq c, y \in \mathbb{Q}^m\}$. We partition the columns of A as $(B \,|\, N)$ for a nonsingular, square matrix $B \in \mathbb{Q}^{m \times m}$ and x into $(x_B \,|\, x_N)$. The standard form system brought to basic form reads $Bx_B + Nx_N = b$, i.e., $x_B = B^{-1}b - B^{-1}Nx_N$. Substituting for x_B, the objective function becomes $c_Bx_B + c_Nx_N = c_BB^{-1}b + (c_N - c_BB^{-1}N)x_N$. Letting $x_N = 0$, we obtain the corresponding basic solution $x_B = B^{-1}b$, $x_N = 0$, of value $c_BB^{-1}b$. When looking at the problem from the current basis B, the vector of *reduced costs*, i.e., the new vector of objective function coefficients for the nonbasic variables (or, stated differently, the derivative of the former w.r.t. each of the latter ones), reads $\bar{c}_N := c_N - c_BB^{-1}N$. The basis B yields an optimal solution if and only if $\bar{c}_{N_j} \geq 0$ for all nonbasic variables of index j.

It is easily shown that, if B is optimal, $y^* := c_BB^{-1}$ is an optimal solution to the dual D. To observe this, partition the dual system $yA \leq c$ into $yB \leq c_B$ and $yN \leq c_N$. By construction, $y^*B \leq c_B$ becomes $c_BB^{-1}B \leq c_B$, thus being trivially satisfied. By substitution, $yN \leq c_N$ becomes $c_BB^{-1}N \leq c_N$, i.e., $c_N - c_BB^{-1}N \geq 0$. Since $c_N - c_BB^{-1}N = \bar{c}_N$ and $\bar{c}_N \geq 0$ by the assumption of optimality of B, y^* satisfies $yN \leq c_N$. This allows to rewrite $\bar{c}_N$ as $\bar{c}_N = c_N - y^*N$. The jth component reads $\bar{c}_{N_j} = c_{N_j} - y^*N_{\cdot j}$, where $N_{\cdot j}$ is the jth column of N. From a dual point of view, $\bar{c}_{N_j}$ amounts to, if negative, the violation of the jth constraint $yN_{\cdot j} \leq c_{N_j}$ at y^*. Thus, the correspondence between cut violation and dual reduced costs is formally made clear.

When considering a cutting plane method to solve $P_{\mathcal{C}}$ (as defined in Section 1), D becomes the primal problem, to which rows are added. The previous derivations can be straightforwardly applied after introducing the sign constraints $y \geq 0$ into the system $yA \leq c$ of D. Nonbasic columns $N_{\cdot j}$ of P (which are rows of D) are now the inequalities $\alpha x \leq \alpha_0$ in $\mathcal{C}$. Thus, solving D with the standard cutting plane method which introduces a maximally violated inequality at a time precisely amounts to solving P adopting Dantzig's pivoting rule. Hence, according to the quotation from [Har73], we can expect the standard cutting plane method to be as inefficient as the naive version of the simplex method employing reduced-cost pivoting.

2.2 An alternative from the simplex method literature: steepest-edge pricing

In the simplex method, each reduced cost $\bar{c}_{N_j}$ provides the exact improvement per unit increase of the nonbasic variable x_{N_j} only if the edge η_j of the polyhedron on which we move when pivoting on such variable is parallel to one of the axes. An alternative yielding the exact unit improvement in the general case is given by the so-called *steepest-edge pricing* rule, see [GR77] and [FG92]. Rather than reduced-costs, the rule considers the directional derivative of the objective function c along η_j, which is obtained by a simple projection of c onto η_j. According to [Bix02] and [BGRW04], such pricing rule (when efficiently implemented) is the most important reason why "implementations of the dual simplex algorithm have become so powerful".

More formally, assume that a feasible (but not optimal) solution x to P is given. Let us assume a nondegenerate case where we pivot on a nonbasic variable x_{N_j}, increasing it from its nonbasic value of 0 to $\theta > 0$. Given the current standard form representation of the problem, i.e., $x_B + B^{-1}Nx_N = B^{-1}b$, by increasing x_{N_j} to θ, x_N becomes $e_j\theta$, where e_j is the unit vector with $n - m$ components and a nonzero jth entry. The standard form system becomes $x_B + B^{-1}Ne_j\theta = B^{-1}b$, from which we deduce that x_B is updated, from $x_B = B^{-1}b$, to $x_B = B^{-1}b - B^{-1}N_{\cdot j}\theta$. From a polyhedral point of view, this corresponds to moving from the current solution x onto the direction $\eta_j := (-B^{-1}N_{\cdot j} \,|\, e_j)$ with a step-length of θ.

By basic trigonometry, the *steepness* of the edge η_j (projection of c onto η_j) can be obtained as $s_j := \dfrac{c\,\eta_j}{||\eta_j||_2}$. In steepest-edge pricing, pivoting is thus carried out on the column yielding the smallest s_j. Figure 1 illustrates the difference between such rule and reduced-cost pivoting and the limits of the two.

Let us consider the cutting plane point of view. Although steepest-edge pricing has been used in the context of column generation methods, see [Lüb10] and the references therein, it has not yet been adapted, to our knowledge, to the context of cutting plane generation. We provide such an adaptation, albeit straightforward, in the following. When pivoting on the column $N_{\cdot j}$ of P, a new cutting plane $yN_{\cdot j} \leq c_{N_j}$ is introduced into D. With the notation of Section 1, this is equivalent to introducing the cut $\alpha x \leq \alpha_0$ into $P_{\mathcal{C}}$. In the expression for the steepness measure s_j, the numerator $c\,\eta_j$ reads $c\,(-B^{-1}N_{\cdot j} \,|\, e_j) = -c_B B^{-1}N_{\cdot j} + c_{N_j}$. When considering the dual, since $y^* = c_B B^{-1}$, we obtain $-y^* N_{\cdot j} + c_{N_j}$ which, if $s_j < 0$, is equal to the violation of the jth constraint. Adopting again the notation of Section 1, where $N_{\cdot j} = \alpha$, the numerator reads $\alpha_0 - x^*\alpha$ whereas, in the denominator, we have $\|\eta_j\|_2 =$

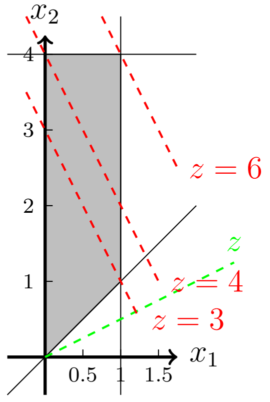

Figure 1: Consider the problem $z = \max\{2x_1 + x_2 : x_1 - x_2 \leq 0, x_1 \in [0,1], x_2 \in [0,4]\}$. Starting from $(0,0)$, both x_1 and x_2 are nonbasic, with reduced costs $\bar{c}_1 = 2, \bar{c}_2 = 1$. With reduced-cost pricing, we pivot on x_1, increasing it by 1 ($\theta = 1$). Due to moving the solution from $(0,0)$ (value $z = 0$) to $(1,1)$ (value $z = 3$), we have an improvement of 3, although $\bar{c}_1\theta = 2$ (reduced-cost estimation). With steepest-edge pricing, the edges are $\eta_1 = (1,1)$ and $\eta_2 = (0,4)$, with steepness $s_1 = \frac{3}{\sqrt{2}}$ and $s_2 = 1$. By pivoting on x_1, we move onto the direction of η_1 by $\theta = \sqrt{2}$ units, reaching $(1,1)$, with an improvement of 3, with $s_1\theta = 3$. Unfortunately, regardless of which rule we use, the best-improving pivot is that on x_2, leading to $(0,4)$ (value $z = 4$).

$\sqrt{\left\|B_D^{-1}N_{\cdot j}\right\|_2^2 + 1}$, which is an increasing function of the Euclidean norm of α induced by the inverse of the optimal basis matrix B_D of the dual. We refer to a cut generated by maximizing the steepness measure of the corresponding dual column as a *dual-steepest-edge cut*. Formally, we can obtain it by minimizing s_j or, equivalently, maximizing $-s_j$. In the latter version, we obtain the following cut generating problem:

$$\max \frac{x^*\alpha - \alpha_0}{\sqrt{\left\|B_D^{-1}\alpha\right\|_2^2 + 1}} \tag{1a}$$

$$\text{s.t.} (\alpha, \alpha_0) \in \mathcal{C}. \tag{1b}$$

Unfortunately, as we will mention in Section 5, solving Problem (1a)–(1b) is typically very hard in practice.

By the aforementioned correspondence between cut generation and pivoting, one would expect, by generating dual-steepest-edge cuts rather than maximally violated ones, an improvement similar to that of steepest-edge pricing in the simplex method. This is hinted at by our computational experiments, as reported in Section 5.

2.3 Coordinated cutting planes as an alternative from the cutting plane literature

As we briefly mentioned in Section 1, many successful implementations of a cutting plane algorithm, both in a pure setting, such as [BCC93] and [ACF07], as well as in a branch-and-cut one, see [BCC96] and [Ach09], adopt the following two-phase technique. First, many cutting planes are gathered by solving a separation problem where the cut violation is maximized (either to optimality or, more often, heuristically). Then, a so-called *cut selection* procedure is used to identify a subset of promising inequalities by considering different cut quality measures. The most frequently used ones include the Euclidean distance between the facet currently induced by the cut into the relaxation and the current infeasible solution, the sparsity of the cut (usually for numerical reasons), and, most importantly, a measure of parallelism w.r.t. the previously generated cuts. Being employed to discard cuts which are too similar to those that were previously found, the latter measure introduces a form of *diversity* among the cutting planes. In the context of a branch-and-cut algorithm such as SCIP, see [Ach09], the cut selection procedure is also guided by many practical reasons such as, e.g., managing the trade-off between the tightness of the bounds and the size of the LP relaxations, so to avoid spending an excessively large computing time in each branch-and-cut node, or keeping the numerical error under control.

More recently, a number of papers have proposed the adoption of *diversification* strategies, often reminiscent of the way cut parallelism is used in cut selection, even in the context of a pure cutting plane method, and in a more direct way. Indeed, computational experience shows that the generation of a diversified set of cuts usually allows for a faster convergence, as addressed, e.g., in [FL07, BS08], and [ACG12]. Differently from the many cases where the practice of generating diverse cuts is addressed heuristically, the authors of [ACG12] address it in a formally precise way, proposing the so-called *coordinated cutting plane generation.* Exploiting the (often natural) presence of many maximally violated inequalities, their method looks for a maximally violated one which also maximizes a diversity measure between itself and the average of the previously generated cuts. As diversity measure, the authors exploit the 1-norm distance, which is a linear function for cuts with binary coefficients. They show that, for inequalities with binary coefficients and a constant right-hand side, a *coordinated cutting plane* can be found by just changing the primal solution x^* that is about to be separated in an appropriate way. More precisely, for a family of cutting planes with $\alpha \in \{0,1\}^n$ and a constant α_0 (that is, which take a given, constant value throughout the family), a coordinated cut can be found by solving, for a sufficiently small constant $\epsilon > 0$, the problem:

$$\max \alpha x^* - \alpha_0 + \epsilon \left\|\alpha - \bar{\alpha}\right\|_1 \tag{2a}$$

$$\text{s.t.} (\alpha, \alpha_0) \in \mathcal{C}, \tag{2b}$$

where $\bar{\alpha} \in [0,1]$ is, component wise, the average of the left-hand sides of the previously generated cuts. For $\alpha \in \{0,1\}^n$, the function $\left\|\alpha - \bar{\alpha}\right\|_1$ can be rewritten as the linear function $(e - 2\bar{\alpha})\alpha + e\bar{\alpha}$, where e is the all-one vector. Overall, this amounts to separating the new point $\hat{x} := x^* + \epsilon(e - 2\bar{\alpha})$ rather than x^*.

3 Bound-optimal cutting planes

In this section, we outline our main contribution. Inspired by Dantzig's "greatest change" criterion, we propose a cut generating problem capable of producing one or more cutting planes which, when introduced into the current relaxation, yield the largest possible bound improvement.

Formally, we introduce the following:

Definition 1 (Bound-optimal cutting plane)**.** *Consider an integer program P, its continuous relaxation P_{LP} =* $\max\{cx : Ax \leq b, x \in \mathbb{Q}^n\}$*, and a family of valid inequalities $\mathcal{C}$. For*

$A \in \mathbb{Q}^{m\times n}$ and $b \in \mathbb{Q}^m$, assume that $Ax \leq b$ contains lower and upper bounds on the components of x as well as any previously introduced valid inequality from $\mathcal{C}$. Let x^* be the corresponding continuous solution yielding a bound of z_{LP}. A bound-optimal cutting plane is a valid inequality $\alpha x \leq \alpha_0$ in $\mathcal{C}$ which maximizes the bound improvement $z_{LP} - z'_{LP}$, where z'_{LP} is the optimal value of P_{LP} after $\alpha x \leq \alpha_0$ has been added to it.

In the rest of the section, we first show how to construct a mathematical program which generates a bound-optimal cutting plane. Then, we extend the definition to the generation of k cuts which, jointly, yield the largest bound improvement. We conclude by discussing some important properties of our bound-optimal cutting plane method, also comparing it to other cut generation techniques.

3.1 Generation of bound-optimal cutting planes

With the following theorem, we show how a bound optimal cutting plane as defined in Definition 1 can be derived via mathematical programming:

Theorem 1. *Assuming that $\mathcal{C}$ can be represented as a mathematical program, a bound-optimal cutting plane as in Definition 1 can be found by solving the following problem:*

$$\min \sum_{j=1}^{n} c_j x_j \tag{3a}$$

$$\text{s.t.} \sum_{j=1}^{n} a_{ij} x_j \leq b_i \qquad \forall i = 1, \dots, m \tag{3b}$$

$$\sum_{j=1}^{n} \alpha_j x_j \leq \alpha_0 \tag{3c}$$

$$x_j \geq 0 \qquad \forall j = 1, \dots, n \tag{3d}$$

$$\sum_{i=1}^{m} a_{ij} y_i + \alpha_j y_{m+1} \geq c_j \qquad \forall j = 1, \dots, n \tag{3e}$$

$$y_i \geq 0 \qquad \forall i = 1, \dots, m+1 \tag{3f}$$

$$\sum_{j=1}^{n} c_j x_j = \sum_{i=1}^{m} b_i y_i + \alpha_0 y_{m+1} \tag{3g}$$

$$(\alpha, \alpha_0) \in \mathcal{C}. \tag{3h}$$

Proof. By (3b)–(3d), P'_{LP} is rewritten as an LP of variable $x \in \mathbb{R}^n$, which is a parametric LP due to the dependence on the coefficients (α, α_0) of the new cut $\alpha x \leq \alpha_0$ (which are variables in this case). The validity of the cut is imposed in (3h). The optimality of x for P'_{LP} is enforced by introducing the dual of P'_{LP} in (3e)–(3f), which is a parametric LP of variable $y \in \mathbb{R}^{m+1}$. The variable y_{m+1} is the dual variable of the new inequality in (3c). The strong LP duality relationship imposed in (3g) further links the two problems, yielding a primal-dual optimal pair. Since, for any given $(\alpha, \alpha_0) \in \mathcal{C}$, every x which is feasible for (3b)–(3h) is optimal for P'_{LP}, a bound-optimal cut is thus found by minimizing the objective function of P'_{LP}, rather than by maximizing it. $\square$

Problem (3a)–(3h), which is of polynomial size w.r.t. m and n (the numbers of, respectively, constraints and variables in the problem), is quadratic and nonconvex because of the bilinear products $\alpha_j x_j$ in (3c), $\alpha_j y_{m+1}$ in (3e), and $\alpha_0 y_{m+1}$ in (3g). To arrive at a MILP formulation, we exploit the well-known McCormick envelope of the bilinear product of two (different) variables, see [McC76]. Consider two continuous bounded variables $x \in [x^L, x^U]$, $y \in [y^L, y^U]$. The bilinear feasible set $\{(z, x, y) \in \mathbb{R}^3 : z = xy\}$, which is a nonconvex nor concave surface, admits a unique and compact convex envelope which is a polyhedron described by the following four inequalities:

$$z \geq x^L y + x y^L - x^L y^L \tag{4}$$
$$z \geq x^U y + x y^U - x^U y^U \tag{5}$$
$$z \leq x^L y + x y^U - x^L y^U \tag{6}$$
$$z \leq x^U y + x y^L - x^U y^L. \tag{7}$$

Due to its pointwise minimality, pointed out in [AKF83], the envelope meets the nonlinear surface defined by $z = xy$ at its borders. Therefore, whenever one of the two variables takes binary values, the envelope amounts to a reformulation of the bilinear product xy rather than to a relaxation. By linearizing each bilinear product in Problem (3a)–(3h), the following result is directly obtained:

Corollary 1. *If $\alpha \in \{0,1\}^n$ for all $j = 1, \ldots, n$ and α_0 is an affine function of α, provided that $\mathcal{C}$ can be expressed as a mixed-integer set, then problem* (3a)–(3h) *can be cast as a Mixed 0-1 Integer Program of polynomial size in m, n.*

Proof. The products $\alpha_j x_j$ and $\alpha_j y_{m+1}$ can be directly linearized via the McCormick envelope. If $\alpha_0 = \sum_{j=1}^n w_j \alpha_j + w_0$ for some $w_0, \ldots, w_n \in \mathbb{Q}$, the product $\alpha_0 y_{m+1}$ in (3g) becomes $\sum_{j=1}^n w_j \alpha_j y_{m+1} + w_0 y_{m+1}$ and, thus, the McCormick envelope can be directly applied to each product $w_j \alpha_j y_{m+1}$. $\square$

This result encompasses many families of combinatorial cutting planes, such as clique or stable set inequalities (where $\alpha_0 = 1$), cut-set inequalities (where $\alpha_0 = 1$ for connectivity or $\alpha_0 = 2$ for biconnectivity), and knapsack cover inequalities (where $\alpha_0 = \sum_{j=1}^n \alpha_j - 1$), which are valid for many integer programming problems. See [NW88] and the references therein.

Note that, in theory, the corollary can be extended to any bounded $(\alpha, \alpha_0) \in \mathbb{Z}^{n+1}$. Indeed, assuming $\alpha_j \leq M$ for all $j = 1, \ldots, n$, and $\alpha_0 \leq M$, for some $M \in \mathbb{Q}^+$, it suffices to rewrite each variable α_j and α_0 as the sum of at most M 0-1 variables, thus transforming all the bilinear products between an integer and a continuous variable in (3c), (3e), and (3g) into the sum of M bilinear products between a binary and a continuous variable. Each of them can then be rewritten in a linear fashion by employing the McCormick envelope. Unfortunately, in the general case this procedure yields a Mixed 0-1 Integer Program of a size which is polynomial w.r.t. m and n, but pseudopolynomial w.r.t. M –a problem which is likely too hard to be solved in practice. The size of the problem can be substantially reduced by binary encoding, namely, by rewriting each variable α_j as $\alpha_j = \sum_{i=0}^{\log M} 2^i z_i$, where $z \in \{0,1\}^{\log M+1}$, although the problem might still be excessively large in practice.

3.2 Extension to k bound-optimal cuts at a time

One of the most interesting features of bound-optimal cutting planes is that a slight modification of Problem (3a)–(3h) allows for the generation of *any* number k of cuts which, jointly, yield the largest bound improvement. Formally:

Definition 2 (k Bound-optimal cutting planes)**.** *Consider the continuous relaxation P_{LP} as in Definition 1, admitting an optimal solution of value z_{LP}. A set of k bound-optimal cutting planes is a collection of k valid inequalities $\alpha^h x \leq \alpha_0^h$ in $\mathcal{C}$, for $h = 1, \ldots, k$, which jointly maximize the bound improvement $z_{LP} - z'_{LP}$, where z'_{LP} is the optimal value of P_{LP} after $\alpha^h x \leq \alpha_0^h$, for all $h = 1, \ldots, k$, have been added to it.*

We can extend Problem (3a)–(3h) to generate such a set of k bound-optimal cutting planes, obtaining the following:

$$\min \sum_{j=1}^{n} c_j x_j \tag{8a}$$

$$\text{s.t.} \sum_{j=1}^{n} a_{ij} x_j \leq b_i \qquad \forall i = 1, \ldots, m \tag{8b}$$

$$x_j \geq 0 \qquad \forall j = 1, \ldots, n \tag{8c}$$

$$\sum_{j=1}^{n} \alpha_j^h x_j \leq \alpha_0^h \qquad \forall h = 1, \ldots, k \tag{8d}$$

$$\sum_{i=1}^{m} a_{ij} y_i + \sum_{h=1}^{k} \alpha_j^h y_{m+h} \geq c_j \qquad \forall j = 1, \ldots, n \tag{8e}$$

$$y_i \geq 0 \qquad \forall i = 1, \ldots, m+k \tag{8f}$$

$$\sum_{j=1}^{n} c_j x_j = \sum_{i=1}^{m} b_i y_i + \sum_{h=1}^{k} \alpha_0^h y_{m+h} \tag{8g}$$

$$(\alpha^h, \alpha_0^h) \in \mathcal{C} \qquad \forall h = 1, \ldots, k. \tag{8h}$$

Problem (8a)–(8h) can be linearized similarly to Problem (3a)–(3h). The impact of generating k bound-optimal cuts at a time is highlighted in the following.

3.3 The bound-optimal cutting plane method

Given a positive integer k, the *bound-optimal cutting plane method* amounts to iterating over two steps:

1) Solve Problem (3a)–(3h) for $k = 1$ or Problem (8a)–(8h) for $k \geq 2$;

2) Add the k new inequalities $\alpha^h x \leq \alpha_0^h$, for $h = 1, \ldots, k$, to $Ax \leq b$,

until Step 1 yields a solution of 0 objective value.

Differently from a standard cutting plane algorithm, the bound-optimal cutting plane method does not require to reoptimize the LP relaxation, which is *embedded* in the separation problem and automatically reoptimized in Step 1. Secondly, the bound-optimal cutting plane method does not involve the concept of *separation* of an infeasible solution x^*. The new optimal solution of the LP relaxation, obtained when the cuts are added to it, is represented only implicitly in the cut generating problem by means of strong duality, wheres the previous infeasible solution x^* is entirely not needed.

Most importantly, and differently from other cutting plane algorithms, the bound-optimal cutting plane method does not account for cut violation at all. Note that the generation of a cut with a strictly positive cut violation suffices to guarantee the convergence of a cutting plane method. It is the case both of dual-steepest-edge cuts (where the steepness of the dual edge is strictly positive if and only if the cut violation is) and of cut coordination, as well as of the standard method.

When adopting bound-optimal cutting planes though, the corresponding cutting plane method *may not* converge to an optimal solution of $P_{\mathcal{C}}$. This is the case of problems where the current LP relaxation admits an optimal facet but, for a given k, $\mathcal{C}$ does not contain any set of k cuts which, jointly, allow to cut the entire facet. An example for $k = 1$ is shown in Figure 2, where the case of case of $k = 2$, for which the method converges, is also illustrated.

As Figure 2 illustrates, the generation of a single cut k times is not equivalent to that of k cuts at the same time. Typically, as we will show in Section 5, the bound at which the bound-optimal cutting plane method halts becomes better for larger values of k. This is because, for a larger k, there are more chances of finding a set of k cuts which simultaneously allow to entirely cut the optimal facet of the current relaxation, thus preventing the issue shown in the figure.

Note that, in principle, any problem $P_{\mathcal{C}}$, regardless of the number of inequalities contained in $\mathcal{C}$, can be solved in a single iteration of the bound-optimal cutting plane method by solving Problem (8a)–(8h) to optimality for $k = n$. This is because, for any problem in $\mathbb{Q}^n$, exactly n inequalities are needed to uniquely identify a vertex and, hence, no more than n cutting planes from the family $\mathcal{C}$ are needed. Unfortunately, computational experiments show that Problem (8a)–(8h) is extremely hard to solve for large values of k whereas, for lower values, it can still be solved in a reasonable amount of computing time.

To easily obtain a fast and converging method, a lucrative option is that of coupling the bound-optimal cutting plane method with another one where strictly violated cutting planes are introduced. We will experiment with such a technique in Section 5.

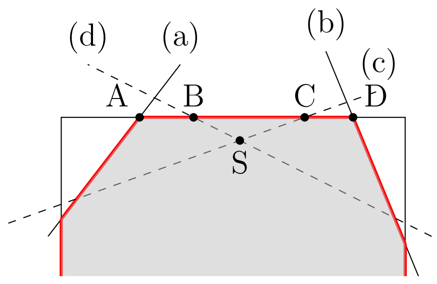

Figure 2: Consider a problem $P_{\mathcal{C}}$ where $\mathcal{C}$ contains four cuts, two of which, (a) and (b), have already been added to the LP relaxation. Assume that the objective function gradient is orthogonal to the current facet AD. The primal bound cannot be improved by introducing any single cut because, if cut (c) (resp. cut (d)), is added, the segment $CD \in AD$ (resp. $AB \in AD$) remains feasible. As a consequence, the bound-optimal cut generation problem for $k = 1$, as defined in (3a)–(3h), admits four optimal solutions (all with objective function 0), namely the four cuts (a), (b), (c), and (d), regardless of the fact that cuts (a) and (b) have already been generated. Hence, the method does not converge to the optimal solution S of $P_{\mathcal{C}}$ for $k = 1$. On the contrary, it does when solving (8a)–(8h) for $k = 2$, where cuts (c) and (d) are simultaneously generated.

4 Case study

In this section, we show an adaptation of bound-optimal cutting planes to the problem of finding the *fractional clique number* (FCN) of a graph, which we will consider in Section 5 for our computational experiments.

4.1 Adaptation to the fractional clique number problem

Given an undirected graph $G = (V, E)$, we look for a fractional clique, namely, for a nonnegative real function on V where the sum of its values on the vertices of any stable set in the graph is bounded from above by 1 and where the sum of the values it assigns to the vertices is maximized. This conforms to the definition of $P_{\mathcal{C}}$ that is given in Section 1. The problem amounts to an LP relaxation of the *maximum clique* problem where we look for a clique in G (a complete subgraph) of maximum cardinality. Denoting by $\mathcal{S} := \{S_1, \ldots, S_m\}$ the set of all (maximal) stable sets of G, the FCN problem can be solved via the following LP with, in the general case, exponentially many constraints:

$$\max \sum_{j \in V} x_j \tag{9a}$$

$$\text{s.t.} \sum_{j \in S_i} x_j \leq 1 \qquad \forall S_i \in \mathcal{S} \tag{9b}$$

$$x_j \geq 0 \qquad \forall j \in V. \tag{9c}$$

For simplicity, we will assume (also in the computational experiments) that, at the first iteration, only the trivial stable sets $S_j = \{j\}$, for all $j \in V$, are present in the relaxation. The dual of Problem (9a)–(9c), which is later used to construct the program which generates a bound-optimal cutting plane, reads:

$$\min \sum_{S_i \in \mathcal{S}} y_i \tag{10a}$$

$$\text{s.t.} \sum_{S_i \in \mathcal{S}: j \in S_i} y_i \geq 1 \qquad \forall i \in V \tag{10b}$$

$$y_i \geq 0 \qquad \forall i \in \mathcal{S}. \tag{10c}$$

It amounts to an LP relaxation of the graph coloring problem where each vertex has to be covered (fractionally) by stable sets.

Let x^* be the current solution to the relaxation of (9a)–(9c) containing only the stable set inequalities generated so far. The standard separation problem looking for a maximally violated stable set inequality can be cast as the following 0-1 IP:

$$v = \max \sum_{j \in V} x_j^* \alpha_j \tag{11a}$$

$$\text{s.t.} \alpha_i + \alpha_j \leq 1 \qquad \forall \{i, j\} \in E \tag{11b}$$

$$\alpha_j \in \{0, 1\} \qquad \forall j \in V. \tag{11c}$$

A valid violated inequality is found for any solution where $v > 1$. Otherwise, any optimal solution where $v \leq 1$ certifies that x^* is an optimal solution to the FCN. A dual-steepest-edge cutting plane can be found by solving the following 0-1 Nonlinear Program:

$$\max \frac{\sum_{j \in V} x_j^* \alpha_j - 1}{\sqrt{\left\| B_D^{-1} \alpha \right\|_2^2 + 1}} \tag{12a}$$

$$\text{s.t.} \alpha_i + \alpha_j \leq 1 \qquad \forall \{i, j\} \in E \tag{12b}$$

$$\alpha_j \in \{0, 1\} \qquad \forall j \in V, \tag{12c}$$

where B_D^{-1} is the inverse matrix of an optimal solution to the restriction of (10a)–(10c) containing only the columns corresponding to the stable sets generated so far.

Assume that m stable set inequalities have been generated. Applying Theorem 1 and Corollary 1, a bound-optimal stable set inequality is obtained as a solution to the following

Mixed 0-1 IP:

$$\min \sum_{j \in V} x_j \tag{13a}$$

$$\text{s.t.} \sum_{j \in S_i} x_j \leq 1 \qquad \forall i = 1, \dots, m \tag{13b}$$

$$\sum_{j \in V} \alpha_j x_j \leq 1 \tag{13c}$$

$$x_j \geq 0 \qquad \forall j \in V \tag{13d}$$

$$\sum_{i=1: S_i \ni j}^{m} y_i + \alpha_j y_{m+1} \geq 1 \qquad \forall j \in V \tag{13e}$$

$$y_i \geq 0 \qquad \forall i = 1, \dots, m+1 \tag{13f}$$

$$\sum_{j \in V} x_j = \sum_{i=1}^{m} y_i + y_{m+1} \tag{13g}$$

$$\alpha_i + \alpha_j \leq 1 \qquad \forall \{i, j\} \in E \tag{13h}$$

$$\alpha_j \in \{0, 1\}^n \qquad \forall j \in V. \tag{13i}$$

The problem is linearized, for each $j = 1, \dots, n$, by substituting two new variables $z_j \geq 0$ and $h_j \geq 0$ for, respectively, $\alpha_j x_j$ and $\alpha_j y_{m+1}$. The linearization for each $j \in V$ contains the following six McCormick envelope constraints:

$$z_j \geq x_j + \alpha_j - 1 \tag{14a}$$

$$z_j \leq x_j \tag{14b}$$

$$z_j \leq \alpha_j \tag{14c}$$

$$h_j \geq y_{m+1} + \alpha_j - 1 \tag{14d}$$

$$h_j \leq y_{m+1} \tag{14e}$$

$$h_j \leq \alpha_j. \tag{14f}$$

Due to the direction of the Constraints (13c) and (13e) and the nonnegativity of the variables α_j and y_{m+1}, the Constraints (14b), (14c), and (14d) can be dropped. The extension to the case of k bound-optimal cuts is straightforward.

4.2 A note on the issue of cut domination

Stable set inequalities are facet-defining for problem (9a)–(9c) (and, thus, not dominated by other stable set inequalities) if and only if they are inclusion-wise maximal, see for instance [NW88]. From a practical point of view, ensuring that the cuts that are generated are nondominated is paramount to have a successful cutting plane algorithm. In [ACG12], the maximality of the inequalities is enforced via a simple technique by which, rather than looking for a stable set S with an incidence vector α maximizing $\sum_{j \in V} x_j^* \alpha_j$, we look for one which maximizes $\sum_{j \in V}(x_j^* + \epsilon)\alpha_j$ for a sufficiently small ϵ. This way, given any nonmaximal stable set S_1, any other stable set S_2 which is a superset of the previous one will have a larger objective function value, thus preventing the generation of the inequality for S_1. In the paper, the authors describe how to find a suitable value for ϵ which guarantees that any solution to their separation problem be lexicographically optimal w.r.t., first, the cut violation and, then, the cardinality of the stable set.

In this paper, rather than modifying the objective function, we propose a family of inequalities which directly guarantees the stable set maximality. Given any stable set S with incidence vector $\alpha \in \{0,1\}^n$, S is maximal if and only if, for each vertex $j \in V \setminus S$, at least one of the vertices $i \in \delta(j)$, where $\delta(j) := \{\{v,w\} \in E : v = j\}$, is in S. This is because, if it is not the case, then S is not maximal since $S \cup \{j\}$ is a valid stable set containing S. This is reflected by the following constraint:

$$\sum_{\{i,j\} \in \delta(j)} \alpha_i \geq 1 - \alpha_j \qquad \forall j \in V. \tag{15}$$

5 Computational experiments

In this section, we evaluate the impact of the bound-optimal cutting plane method on the FCN problem. Computational experiments are carried out to compare our cutting plane paradigm to the other algorithms discussed in the paper, aiming at assessing their difference w.r.t. the total number of iterations and/or cutting planes required to achieve a certain bound. As we mentioned, we refrain from resorting to heuristics so to obtain a clearer picture of the impact of the different techniques.

We consider a set of 39 instances taken from the second DIMACS challenge on max clique, coloring, and satisfiability, see [JT96], and from the Treewidth library, described in [BvdB04]. Throughout the data set, complemented instances are denoted with a name ending in `c`. The instance `petersen_t` is equivalent to `petersen` with different vertex labels. All the experiments are carried out with CPLEX 12.4, adopting AMPL as modeling language. They are run on three identical machines equipped with 3.40GHz Intel i7-3770 CPUs and 32GB of RAM, using a single thread.

We consider the following cutting plane algorithms:

- STD: standard generation of a maximally violated inequalities;
- COORD: generation of coordinated cutting planes, as in [ACG12];
- STEEP: generation of dual-steepest-edge cutting planes;
- BOC-k: generation of k bound-optimal cuts at time, with $k = 1, 2, 3$;
- BOC-k-COORD: generation of k bound-optimal cuts at time, with $k = 1, 2, 3$; convergence to an optimal solution to the FCN problem is guaranteed by generating coordinated cutting planes after the bound-optimal cutting plane method halts.

In all cases, except for BOC-k and BOC-k-COORD for $k \geq 2$, we generate a single cutting plane at a time. For BOC-k, a time limit of 7200 seconds (two hours) is imposed. For BOC-k-COORD, in order to always achieve an optimal solution, coordinated cutting planes are generated even if the time limit is exceeded.

Unfortunately, solving the dual-steepest-edge cut generating problem as stated in (1a)–(1b) with state-of-the-art mixed-integer nonlinear solvers like SCIP turns out to be computationally prohibitive for most of the instances. Similarly, a reformulation which gets rid of the nonconvexities in the problem (except for its discreteness) suffers from major numerical issues and premature convergence to not optimal solutions. As a consequence, in the following experiments for STEEP we resort to an *a priori* enumeration of all the maximal stable sets in the graph via the algorithm proposed in [BK73], as implemented in the Boost Graph Library, see [SLL01]. At each iteration, the cut maximizing the dual-steepest-edge measure is simply found by evaluating the measure on all the not yet introduced cuts and selecting that yielding the largest value. Although inefficient, this technique permits to tackle more than half of the instances (all those with, roughly, less than 3000 maximal stable sets), thus allowing for some interesting comparisons.

In all cases, the inequalities that we generate are guaranteed to be nondominated by means of Constraint (15). For STEEP, the maximality is guaranteed by construction via the Bron-Kerbosch algorithm.

In the upcoming tables, for each instance and algorithm, we report two main figures: the number of iterations (Iter) and the number of cuts (Cuts), the values of which only differ if adopting BOC for $k \geq 2$, and the achieved bound (Bound). Due to the way the experiments are carried out, the latter figure is the same for all the methods. Computing times in seconds are also reported (Time). For STEEP, the latter does not account for the time invested in the enumeration.

To facilitate the comparisons, we also report aggregate figures via the so-called *geometric mean of ratios*. More specifically, let R be the algorithm whose figures are reported in the left-most part of each table. We compare to R each other algorithm A in the table for all of its figures F (Iter, Cuts, and Time) by first computing the ratio, for all the instances, between F for A and F for R. Then, we aggregate such ratios via the shifted geometric mean, with a shift of 0.01. Such mean is then reported, for each figure and algorithm, in the last two lines of each table, considering all the instances in the Aggregate-All line and only those for which a result obtained with STEEP is available in the Aggregate-Partial line.

5.1 Comparisons for a given bound

We first illustrate an important feature of bound-optimal cutting planes, namely, that regardless of the bound at which BOC-k halts, such bound is achieved by our method in substantially less iterations and cuts than any of STD, COORD, and STEEP.

The three Tables 1, 2, and 3, one per value of $k = 1, 2, 3$, report a comparison between BOC-k and STD, COORD, and STEEP (when available) when halting the latter three algorithms as soon as the bound provided by BOC-k is reached. In the last two lines of the three tables, aggregate results are reported when comparing the algorithms w.r.t. BOC-k.

As to the number of iterations, STD, COORD, and STEEP are increasingly slower than BOC for increasing values of k. We register the largest difference for $k = 3$ where, considering all the instances, STD and COORD require, respectively, more than six and more than four times the number of iterations needed by BOC-3 to achieve the bound. For STEEP, again for $k = 3$ but only considering the instances solved by it, the method takes almost four times the number of iterations needed by BOC-3.

STD, COORD, and STEEP also generate more cutting planes than BOC-k, although without a clear trend for increasing values of k. When considering all the instances, STD generates between 1.81 and 2.20 times the number of cuts produced by BOC-k, whereas COORD generates between 1.36 and 1.44 times such number. For STEEP, on the instances solved by it, the number of cutting planes is between 1.21 and 1.33 times that for BOC-k.

As expected, the computing time required to find bound-optimal cutting planes grows very fast for increasingly larger values of k. Without considering STEEP, for which the enumerative procedure is clearly inefficient, BOC-1 is slower than both STD and COORD by a factor of 2 (with both STD and COORD being more than 50% faster than BOC-1). For BOC-3, the difference in running time ranges over two orders of magnitude.

On average, BOC-1 stops at a bound that is, in geometric mean of ratios, 22% larger than the value of an optimal solution to the FCN problem. Such difference decreases to 15% for BOC-2 and to 13% for BOC-3, thus confirming the observation reported in Section 3, namely, that for larger values of k we are more likely to prevent the method from halting prematurely. Note that, even if this holds on average on our computational experiments, there are still cases where, for a larger k, the bound achieved by BOC-k is worse than that for a lower k. This is the case, for instance, of `queen7_7`, for which we obtan an optimal solution of value 7 for $k = 1, 2$ but, for $k = 3$, BOC halts at a worse bound of 8. This is likely due to the presence of many sets of k bound-optimal cutting planes yielding the largest bound variation. Due to such multiplicity of optimal solutions to our cut generating problem, the choice of one such set of cuts over another one typically determines a different sequence of cutting planes, thus influencing the method up to, in some cases, the bound at which it halts.

To better emphasize the substantially smaller number of iterations needed by the bound-optimal cutting plane method to achieve the bound at which it halts, Figure 3 reports a graphical illustration of the typical evolution of the bounds for BOC-k (for $k = 1, 2, 3$), STD, COORD, and STEEP for four instances, as a function of the number of iterations. Overall, COORD and STEEP are mostly comparable, with COORD being faster on the first three instances and STEEP being faster on the last one. In particular, the charts illustrate the large improvement provided by BOC-k, especially in the earlier cutting plane iterations. The bound that it provides is, for all values of k and at every iteration, always tighter than that given by STD and, for $k = 2, 3$, always tighter than that given by any of the other methods.

5.2 Overall comparisons

We now compare BOC-k-COORD (the extension of BOC-k coupled with COORD) to STD, COORD, and STEEP. Overall results are reported in Table 4. In the two final lines which provide aggregate results, comparisons are carried out w.r.t. STD. All problems for which the results are reported are solved to optimality.

On average, on the instances solved by STEEP, COORD and STEEP yield the same reduction of 16% in the number of iterations (and thus of cutting planes) w.r.t. STD. The reduction for COORD when considering all the instances is comparable, being only 2% larger. Recalling the correspondence with pivoting rules, this confirms the experience with the simplex method on the superiority of steepest-edge pricing when compared to reduced-cost pricing. Interestingly, on average, coordinated cutting plane generation matches such an improvement.

Let us now consider BOC-k-COORD. As to the number of iterations, the method is increasingly faster, for increasing values of k, than STD, COORD, and STEEP, with a reduction w.r.t. STD, on average and when considering all the instances, of 22% for $k = 1$, 35% for $k = 2$, and 43% for $k = 3$. The average improvement w.r.t. COORD is of 4%, 18%, and 25% for, respectively, $k = 1, 2, 3$. That w.r.t. STEEP, when only considering the instances which are solved by the latter, is of 9%, 13%, and 32% for, respectively, $k = 1, 2, 3$.

As to the number of cutting planes that are generated, we still have a reduction w.r.t. STD for all values of k, although such reduction decreases for increasing values of k. On average, over all the instances, it is of 22% for $k = 1$, 18% for $k = 2$, and 16% for $k = 3$. It amounts to an improvement w.r.t. COORD of 4% for $k = 1$ whereas, for $k = 2$, the number of cutting planes is, on average, the same and, for $k = 3$, it is slightly increased, by 2%. Similarly, w.r.t. STEEP and only for the instances which it solves, we have an improvement of 9% for $k = 1$ and of 1% for $k = 2$, whereas we observe a slightly larger number, increased by 1%, for $k = 3$. This is likely because, in some iterations, the number of new cutting planes that are active in the new optimal primal solution is strictly smaller than k. As a consequence, the bound-optimal cutting plane method introduces extra cutting planes that are not needed to improve the bound, thus resulting in a number of cuts which is not as small for $k = 2$ and $k = 3$ as that for $k = 1$.

As expected, even in this setting where BOC-k is coupled with COORD, the computing times for BOC-k-COORD are larger than those for STD and COORD. They are of the same order of magnitude for $k = 1$, but roughly 50% larger than those of STD when considering all the instances, and they quickly become much larger for larger values of k, up to, on average, one to two orders of magnitude larger for $k = 3$.

We remark that, while both COORD and STEEP improve over STD w.r.t. the number of iterations and cuts, the improvement given by BOC-k-COORD is, for the appropriate k, larger. When aiming for the smallest number of cutting planes, one would choose BOC-1-COORD, yielding a further improvement of, on average, 4% w.r.t. COORD (when considering all the instances) and of 9% w.r.t. STEEP (when considering those solved by the latter). When aiming for the smallest number of iterations, one would choose BOC-3-COORD, yielding an extra improvement of 24% on the instances solved by STEEP, which goes up to 32% w.r.t. COORD when considering all the instances.

Overall, the results obtained with our bound-optimal cutting plane method for $k = 1$ confirm Dantzig's preference for the "greatest change" rule in the simplex method, as reported in the quotation from [Har73] in Section 1, especially when our method is compared to the standard cutting plane generation based on cut violation. They also show that, regardless of the lack of a theoretical guarantee linking an increased bound improvement per iteration to a reduced total number of iterations, the latter is, in practice, more than favorably affected by the former.

6 Concluding remarks

This work proposed the bound-optimal cutting plane method, a new paradigm for cutting plane generation which produces, at each iteration of the algorithm, k cuts which, jointly, yield the largest bound variation.

We have compared this paradigm to other relevant methods, namely the standard separation of maximally violated inequalities and the generation of a cut corresponding to a steepest-edge in the dual, as well as to the generation of coordinated cutting planes. Experiments on the fractional clique number problem have shown that, with our method, we can obtain a given bound within a much smaller number of iterations when compared to the other techniques, and typically with fewer cuts. While the improvement over the standard cutting plane method is significant, that over the other methods is still substantial. Unfortunately, but as expected, the computing time needed to find bound-optimal cutting planes is typically quite large.

This work suggests that our method might be a promising alternative to the other techniques, hopefully motivating further developments in this direction. Future work includes looking for more efficient ways to solve our cut generating problem, possibly via the development of fast and efficient heuristics (such as, e.g., those based on rounding procedures). When considering the fractional clique number problem, better formulations for the set $\mathcal{C}$ of valid inequalities could be adopted by, e.g., introducing some heuristically generated, nontrivial clique inequalities.

Acknowledgments

This paper is dedicated to the memory of Francesco Maffioli.

The authors are thankful to Stefano Gualandi for many insightful discussions on the topic and to Arie Koster for revising a first draft of the paper. The first author is supported by the Google Focused Grant Program on Mathematical Optimization and Combinatorial Optimization in Europe "Robust Mixed Integer Programming".

		BOC-1			STD			COORD			STEEP		
	Bound	Iter	Cuts	Time	Iter	Cuts	Time	Iter	Cuts	Time	Iter	Cuts	Time
1-FullIns_3	4.00	4	4	0.05	5	5	0.00	4	4	0.00	4	4	39.00
1-FullIns_4	5.00	5	5	0.22	6	6	0.05	5	5	0.04	-	-	-
1-Insertions_4	5.00	4	4	0.12	4	4	0.02	4	4	0.02	-	-	-
2-FullIns_3	5.00	4	4	0.07	7	7	0.03	4	4	0.02	-	-	-
2-Insertions_3	4.00	3	3	0.03	3	3	0.00	3	3	0.00	-	-	-
3-FullIns_3	6.00	5	5	0.16	10	10	0.05	5	5	0.03	-	-	-
3-Insertions_3	4.00	3	3	0.06	3	3	0.00	3	3	0.00	-	-	-
c-fat200-1	13.00	14	14	0.84	67	67	0.39	27	27	0.16	-	-	-
c-fat200-2	24.00	22	22	0.98	70	70	0.76	22	22	0.26	-	-	-
c-fat200-5	84.00	58	58	5.73	164	164	4.21	65	65	1.65	-	-	-
david	12.00	8	8	0.16	21	21	0.12	16	16	0.12	-	-	-
hamming6-2	32.00	32	32	1.03	91	91	1.65	80	80	1.50	48	48	344.00
huck	11.00	9	9	0.11	22	22	0.09	23	23	0.10	-	-	-
jean	10.00	8	8	0.10	25	25	0.10	20	20	0.08	-	-	-
johnson16-2-4	14.00	14	14	15.86	16	16	2.84	16	16	3.51	17	17	473.00
johnson8-2-4	6.00	6	6	0.10	8	8	0.03	8	8	0.03	8	8	24.00
johnson8-4-4	17.00	18	18	3.67	31	31	1.49	24	24	1.20	28	28	120.00
mug88_1	4.00	5	5	0.17	13	13	0.05	4	4	0.01	-	-	-
mug88_25	4.00	4	4	0.14	24	24	0.01	4	4	0.02	-	-	-
myciel3	4.00	3	3	0.02	3	3	0.00	3	3	0.00	3	3	3.00
myciel4	5.00	4	4	0.06	5	5	0.01	5	5	0.01	4	4	17.00
myciel5	6.00	5	5	0.17	5	5	0.02	5	5	0.02	5	5	221.00
petersen	3.00	3	3	0.01	3	3	0.00	3	3	0.00	3	3	2.00
petersen_t	3.00	3	3	0.00	3	3	0.00	3	3	0.00	5	5	3.00
queen5_5	5.00	5	5	0.03	5	5	0.00	5	5	0.00	5	5	15.00
queen6_6	9.00	10	10	0.25	14	14	0.06	12	12	0.05	14	14	208.00
queen7_7	7.00	7	7	0.17	54	54	0.36	31	31	0.16	7	7	585.00
queen8_12	13.00	14	14	0.79	45	45	0.57	39	39	0.49	-	-	-
queen8_8	10.00	10	10	0.73	10	10	0.08	26	26	0.26	-	-	-
queen9_9	11.00	11	11	0.88	38	38	0.38	28	28	0.34	-	-	-
r125.1c	49.00	35	35	33.53	49	49	24.77	50	50	86.48	42	42	117.00
r250.1c	70.00	51	51	704.09	73	73	580.70	81	81	2113.09	-	-	-
sudokuc	9.00	9	9	2.49	16	16	1.14	9	9	0.76	9	9	11.00
ship-shipc	20.00	16	16	1.49	18	18	0.52	27	27	1.04	22	22	28.00
knights8_8c	39.00	25	25	2.92	63	63	2.37	26	26	1.54	49	49	300.00
kneser8-3c	33.00	23	23	5.52	67	67	3.53	23	23	2.43	43	43	459.00
barleyc	20.00	12	12	0.51	23	23	0.48	18	18	0.56	17	17	30.00
alarmc	20.00	9	9	0.28	13	13	0.18	11	11	0.16	10	10	10.00
1ubqc	36.00	21	21	4.18	35	35	2.28	34	34	3.00	25	25	90.00
Aggregate-All		1.00	1.00	1.00	1.81	1.81	0.44	1.36	1.36	0.42	-	-	-
Aggregate-Partial		1.00	1.00	1.00	1.56	1.56	0.49	1.30	1.30	0.48	1.24	1.24	105.83

Table 1: Comparison between BOC-1 and STD, COORD, and STEEP when halting the latter three algorithms as soon as they reach the bound at which BOC-1 halts.

		BOC-2			STD			COORD			STEEP		
	Bound	Iter	Cuts	Time	Iter	Cuts	Time	Iter	Cuts	Time	Iter	Cuts	Time
1-FullIns_3	3.33	4	8	0.40	13	13	0.02	9	9	0.00	15	15	122.00
1-FullIns_4	3.63	11	22	108.87	44	44	1.18	34	34	0.88	-	-	-
1-Insertions_4	5.00	2	4	2.51	4	4	0.02	4	4	0.02	-	-	-
2-FullIns_3	5.00	2	4	0.50	7	7	0.03	4	4	0.02	-	-	-
2-Insertions_3	2.67	4	8	0.95	29	29	0.02	15	15	0.01	-	-	-
3-FullIns_3	6.00	3	6	1.38	10	10	0.05	5	5	0.03	-	-	-
3-Insertions_3	3.00	3	6	1.81	19	19	0.02	6	6	0.00	-	-	-
c-fat200-1	13.00	7	14	3.65	67	67	0.39	27	27	0.16	-	-	-
c-fat200-2	24.00	11	22	3.84	70	70	0.76	22	22	0.26	-	-	-
c-fat200-5	76.00	34	68	19.13	205	205	5.28	128	128	3.28	-	-	-
david	11.00	4	8	0.51	31	31	0.18	22	22	0.17	-	-	-
hamming6-2	32.00	17	34	14.74	91	91	1.65	80	80	1.50	48	48	344.00
huck	11.00	5	10	0.52	22	22	0.09	23	23	0.10	-	-	-
jean	10.00	4	8	0.48	25	25	0.10	20	20	0.08	-	-	-
johnson16-2-4	14.00	7	14	260.30	16	16	2.84	16	16	3.51	17	17	473.00
johnson8-2-4	6.00	3	6	0.36	8	8	0.03	8	8	0.03	8	8	24.00
johnson8-4-4	17.00	8	16	27.21	31	31	1.49	24	24	1.20	28	28	120.00
mug88_1	4.00	2	4	3.10	13	13	0.05	4	4	0.01	-	-	-
mug88_25	4.00	2	4	1.96	24	24	0.01	4	4	0.02	-	-	-
myciel3	4.00	2	4	0.06	3	3	0.00	3	3	0.00	3	3	3.00
myciel4	5.00	2	4	0.24	5	5	0.01	5	5	0.01	4	4	17.00
myciel5	6.00	3	6	1.54	5	5	0.02	5	5	0.02	5	5	221.00
petersen	3.00	2	4	0.04	3	3	0.00	3	3	0.00	3	3	2.00
petersen_t	3.00	2	4	0.04	3	3	0.00	3	3	0.00	5	5	3.00
queen5_5	5.00	3	6	0.18	5	5	0.00	5	5	0.00	5	5	15.00
queen6_6	8.00	5	10	0.78	16	16	0.07	17	17	0.07	16	16	235.00
queen7_7	7.00	4	8	1.82	54	54	0.36	31	31	0.16	7	7	585.00
queen8_12	13.00	7	14	16.81	45	45	0.57	39	39	0.49	-	-	-
queen8_8	10.00	5	10	8.32	10	10	0.08	26	26	0.26	-	-	-
queen9_9	11.00	6	12	11.21	38	38	0.38	28	28	0.34	-	-	-
r125.1c	46.00	19	38	360.79	54	54	26.32	54	54	93.61	53	53	134.00
r250.1c	66.00	30	60	7199.26	100	100	675.94	104	104	2753.35	-	-	-
sudokuc	9.00	5	10	19.71	16	16	1.14	9	9	0.76	9	9	11.00
ship-shipc	19.00	10	20	8.33	28	28	0.77	29	29	1.12	24	24	30.00
knights8_8c	33.00	18	36	46.89	93	93	3.33	51	51	2.66	57	57	338.00
kneser8-3c	29.00	14	28	154.41	86	86	4.40	36	36	3.45	55	55	571.00
barleyc	20.00	7	14	2.90	23	23	0.48	18	18	0.56	17	17	30.00
alarmc	18.00	6	12	0.80	15	15	0.20	16	16	0.23	16	16	13.00
1ubqc	33.00	12	24	46.71	45	45	2.90	39	39	3.43	32	32	109.00
Aggregate-All		1.00	1.00	1.00	3.85	1.93	0.07	2.80	1.40	0.06	-	-	-
Aggregate-Partial		1.00	1.00	1.00	2.99	1.50	0.08	2.53	1.26	0.08	2.41	1.21	17.84

Table 2: Comparison between BOC-2 and STD, COORD, and STEEP when halting the latter three algorithms as soon as they reach the bound at which BOC-2 halts.

		BOC-3			STD			COORD			STEEP		
	Bound	Iter	Cuts	Time	Iter	Cuts	Time	Iter	Cuts	Time	Iter	Cuts	Time
1-FullIns_3	3.33	3	9	1.53	13	13	0.02	9	9	0.00	15	15	122.00
1-FullIns_4	3.67	5	15	459.31	36	36	0.50	24	24	0.29	-	-	-
1-Insertions_4	3.07	7	21	7199.41	45	45	0.20	24	24	0.10	-	-	-
2-FullIns_3	4.25	4	12	6.80	27	27	0.12	17	17	0.07	-	-	-
2-Insertions_3	2.50	6	18	104.47	48	48	0.05	29	29	0.02	-	-	-
3-FullIns_3	5.20	6	18	57.12	36	36	0.20	18	18	0.10	-	-	-
3-Insertions_3	2.67	3	9	234.03	35	35	0.04	14	14	0.01	-	-	-
c-fat200-1	13.00	5	15	21.95	67	67	0.39	27	27	0.16	-	-	-
c-fat200-2	24.00	8	24	18.04	70	70	0.76	22	22	0.26	-	-	-
c-fat200-5	67.00	27	81	722.28	248	248	6.47	189	189	4.85	-	-	-
david	11.00	3	9	4.44	31	31	0.18	22	22	0.17	-	-	-
hamming6-2	32.00	11	33	186.82	91	91	1.65	80	80	1.50	48	48	344.00
huck	11.00	3	9	2.39	22	22	0.09	23	23	0.10	-	-	-
jean	10.00	3	9	2.58	25	25	0.10	20	20	0.08	-	-	-
johnson16-2-4	60.00	2	6	7199.75	5	5	1.31	5	5	1.56	5	5	161.00
johnson8-2-4	4.00	3	9	1.65	8	8	0.03	8	8	0.03	12	12	33.00
johnson8-4-4	14.00	6	18	282.60	56	56	2.64	56	56	2.68	58	58	207.00
mug88_1	3.33	3	9	7199.54	94	94	0.27	21	21	0.05	-	-	-
mug88_25	3.33	3	9	7199.04	58	58	0.04	30	30	0.09	-	-	-
myciel3	3.00	3	9	0.23	9	9	0.00	6	6	0.00	9	9	6.00
myciel4	3.50	3	9	2.18	14	14	0.02	7	7	0.02	13	13	44.00
myciel5	4.00	4	12	15.21	25	25	0.13	15	15	0.08	18	18	720.00
petersen	2.50	2	6	0.08	13	13	0.00	6	6	0.00	14	14	6.00
petersen_t	2.50	2	6	0.09	12	12	0.00	6	6	0.00	5	5	3.00
queen5_5	5.00	2	6	0.39	5	5	0.00	5	5	0.00	5	5	15.00
queen6_6	7.00	5	15	5.61	49	49	0.51	42	42	0.43	48	48	630.00
queen7_7	8.00	4	12	14.94	40	40	0.25	8	8	0.05	7	7	585.00
queen8_12	12.50	6	18	1597.23	55	55	0.74	48	48	0.66	-	-	-
queen8_8	9.33	6	18	906.72	47	47	0.35	38	38	0.39	-	-	-
queen9_9	10.50	5	15	226.86	49	49	0.55	34	34	0.40	-	-	-
r125.1c	46.00	13	39	7197.13	54	54	26.32	54	54	93.61	53	53	134.00
r250.1c	180.00	4	12	7200.84	13	13	269.77	13	13	302.42	-	-	-
sudokuc	9.00	3	9	192.61	16	16	1.14	9	9	0.76	9	9	11.00
ship-shipc	19.00	8	24	112.93	28	28	0.77	29	29	1.12	24	24	30.00
knights8_8c	32.00	12	36	707.48	103	103	3.64	53	53	2.73	58	58	342.00
kneser8-3c	47.00	3	9	7199.41	12	12	0.64	9	9	1.12	11	11	134.00
barleyc	20.00	4	12	17.65	23	23	0.48	18	18	0.56	17	17	30.00
alarmc	18.00	4	12	7.04	15	15	0.20	16	16	0.23	16	16	13.00
1ubqc	32.00	9	27	584.62	48	48	3.09	41	41	3.58	36	36	119.00
Aggregate-All		1.00	1.00	1.00	6.59	2.20	0.01	4.32	1.44	0.00	-	-	-
Aggregate-Partial		1.00	1.00	1.00	5.03	1.68	0.01	3.58	1.19	0.01	3.97	1.33	2.63

Table 3: Comparison between BOC-3 and STD, COORD, and STEEP when halting the latter three algorithms as soon as they reach the bound at which BOC-3 halts.

[h!]

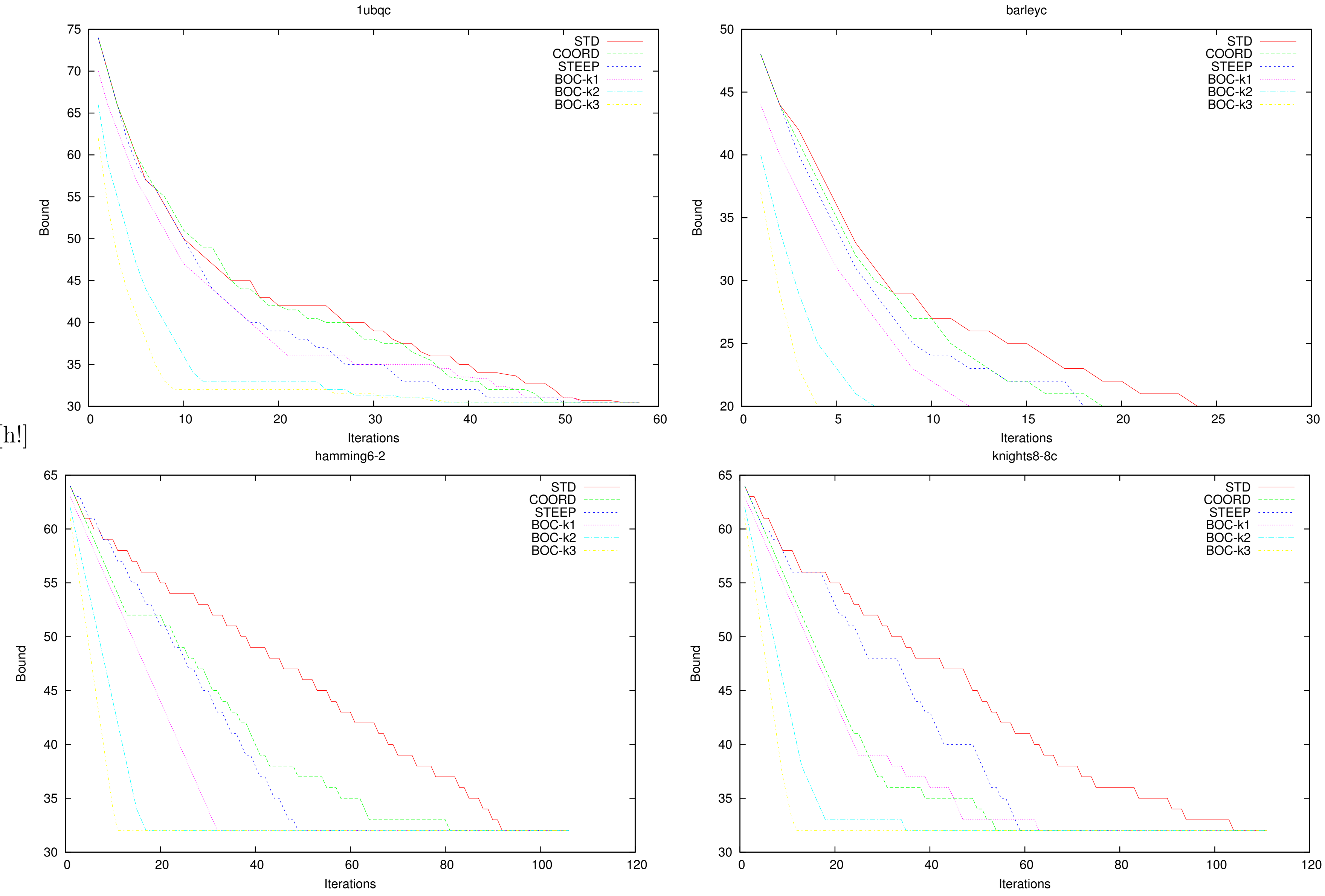

Figure 3: Graphical representation of the bound improvement yielded by BOC, STD, COORD, and STEEP for the instances `1ebqc`, `barleyc`, `hamming-6-2`, and `knights8-8c`, as a function of the number of iterations.

		STD			COORD			STEEP			BOC-1-COORD			BOC-2-COORD			BOC-3-COORD		
	Bound	Iter	Cuts	Time	Iter	Cuts	Time	Iter	Cuts	Time	Iter	Cuts	Time	Iter	Cuts	Time	Iter	Cuts	Time
1-FullIns_3	3.33	13	13	0.02	11	11	0.01	15	15	122.00	14	14	0.07	9	13	0.41	9	15	1.54
1-FullIns_4	3.63	44	44	1.18	34	34	0.88	-	-	-	34	34	1.17	18	29	109.35	25	35	460.06
1-Insertions_4	2.77	218	218	24.27	188	188	22.47	-	-	-	198	198	23.62	197	199	26.99	203	217	7225.08
2-FullIns_3	4.25	27	27	0.12	17	17	0.07	-	-	-	20	20	0.15	16	18	0.59	4	12	6.81
2-Insertions_3	2.42	101	101	2.05	79	79	1.75	-	-	-	85	85	1.82	87	91	2.92	79	91	106.53
3-FullIns_3	5.20	36	36	0.20	19	19	0.11	-	-	-	22	22	0.28	23	26	1.54	9	21	57.14
3-Insertions_3	2.33	202	202	8.57	181	181	8.65	-	-	-	186	186	8.03	167	170	9.58	171	177	241.41
c-fat200-1	12.00	186	186	1.39	166	166	1.35	-	-	-	153	153	1.96	150	157	4.85	139	149	23.01
c-fat200-2	24.00	70	70	0.76	44	44	0.51	-	-	-	65	65	1.48	60	71	4.43	54	70	18.59
c-fat200-5	66.67	254	254	6.62	204	204	5.23	-	-	-	202	202	9.53	166	200	22.56	147	201	725.47
david	11.00	31	31	0.18	24	24	0.18	-	-	-	21	21	0.25	12	16	0.56	18	24	4.55
hamming6-2	32.00	105	105	1.90	82	82	1.53	84	84	545.00	75	75	1.88	56	73	15.50	49	71	187.58
huck	11.00	26	26	0.11	23	23	0.10	-	-	-	18	18	0.15	13	18	0.56	17	23	2.45
jean	10.00	25	25	0.10	20	20	0.08	-	-	-	15	15	0.13	13	17	0.52	11	17	2.62
johnson16-2-4	8.00	20	20	3.44	20	20	5.68	23	23	627.00	22	22	19.67	15	22	263.86	16	20	7203.64
johnson8-2-4	4.00	10	10	0.04	10	10	0.04	12	12	33.00	9	9	0.11	6	9	0.37	3	9	1.65
johnson8-4-4	14.00	56	56	2.64	56	56	2.68	60	60	211.00	56	56	5.56	45	53	29.09	38	50	284.02
mug88_1	3.03	364	364	3.95	270	270	3.08	-	-	-	259	259	3.31	272	274	6.41	268	274	7203.11
mug88_25	3.03	333	333	3.38	256	256	2.34	-	-	-	276	276	2.98	301	303	5.33	293	299	7202.57
myciel3	2.90	14	14	0.01	12	12	0.02	13	13	7.00	14	14	0.03	12	14	0.07	9	15	0.25
myciel4	3.24	33	33	0.44	29	29	0.31	30	30	86.00	29	29	0.40	29	31	0.60	25	31	2.58
myciel5	3.55	77	77	4.64	69	69	4.41	70	70	2431.00	64	64	4.56	62	65	5.84	63	71	20.40
petersen	2.50	13	13	0.00	7	7	0.00	14	14	6.00	6	6	0.01	5	7	0.04	4	8	0.08
petersen_t	2.50	12	12	0.00	7	7	0.00	6	6	4.00	6	6	0.00	7	9	0.04	3	7	0.09
queen5_5	5.00	5	5	0.00	5	5	0.00	5	5	15.00	5	5	0.03	3	6	0.18	2	6	0.40
queen6_6	7.00	49	49	0.51	42	42	0.43	48	48	630.00	41	41	0.54	39	44	1.19	37	47	5.81
queen7_7	7.00	58	58	0.38	37	37	0.21	7	7	585.00	7	7	0.17	20	24	1.91	37	45	15.15
queen8_12	12.00	103	103	1.50	91	91	1.49	-	-	-	99	99	2.04	84	91	18.01	83	95	1598.46
queen8_8	8.44	123	123	3.44	112	112	3.07	-	-	-	115	115	3.81	128	133	12.28	108	120	909.96
queen9_9	9.00	252	252	12.12	256	256	12.83	-	-	-	238	238	13.06	265	271	24.94	244	254	240.88
r125.1c	46.00	54	54	26.32	54	54	93.61	54	54	136.00	51	51	59.72	28	47	375.46	21	47	7209.77
r250.1c	64.00	116	116	725.95	114	114	3045.95	114	114	866.00	106	106	2287.97	80	110	8596.49	93	101	9634.00
sudokuc	9.00	20	20	1.34	22	22	2.08	22	22	19.00	23	23	3.76	18	23	20.89	15	21	193.78
ship-shipc	19.00	29	29	0.80	30	30	1.16	27	27	31.00	27	27	1.95	20	30	8.76	16	32	113.26
knights8_8c	32.00	110	110	3.87	67	67	3.30	77	77	420.00	72	72	4.98	56	74	48.50	43	67	708.92
kneser8-3c	28.00	96	96	4.92	59	59	5.37	76	76	753.00	59	59	8.50	44	58	156.58	53	59	7203.63
barleyc	20.00	27	27	0.55	26	26	0.84	21	21	34.00	16	16	0.61	10	17	2.99	10	18	17.81
alarmc	18.00	19	19	0.23	19	19	0.27	19	19	14.00	17	17	0.39	9	15	0.84	7	15	7.08
1ubqc	30.50	57	57	3.67	49	49	4.20	49	49	148.00	53	53	6.72	40	52	49.02	36	54	586.73
Aggregate-All		1.00	1.00	1.00	0.82	0.82	1.00	-	-	-	0.78	0.78	1.47	0.65	0.82	5.71	0.57	0.84	70.11
Aggregate-Partial		1.00	1.00	1.00	0.84	0.84	1.07	0.84	0.84	194.67	0.75	0.75	1.70	0.61	0.83	7.77	0.52	0.85	54.45

Table 4: Comparison of BOC-k-COORD, for $k = 1, 2, 3$, with STD, COORD, and STEEP.

References

[ACF07] G. Andreello, A. Caprara, and M. Fischetti. Embedding {0, 1/2}-cuts in a branch-and-cut framework: A computational study. *INFORMS Journal on Computing*, 19(2):229–238, 2007.

[ACG12] E. Amaldi, S. Coniglio, and S. Gualandi. Coordinated cutting plane generation via multi-objective separation. *Mathematical Programming*, pages 1–24, 2012.

[Ach09] T. Achterberg. Scip: solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, 2009.

[AKF83] F. Al-Khayyal and J. Falk. Jointly constrained biconvex programming. *Mathematics of Operations Research*, 8(2):273–286, 1983.

[BCC93] E. Balas, S. Ceria, and G. Cornuéjols. A lift-and-project cutting plane algorithm for mixed 0–1 programs. *Mathematical programming*, 58(1-3):295–324, 1993.

[BCC96] E. Balas, S. Ceria, and G. Cornuéjols. Mixed 0-1 programming by lift-and-project in a branch-and-cut framework. *Management Science*, 42(9):1229–1246, 1996.

[BGRW04] R. Bixby, M. Fenelon Z. Gu, E. Rothberg, and R. Wunderling. Mixed integer programming: a progress report. *The Sharpest Cut: The Impact of Manfred Padberg and His Work, MPS-SIAM Series on Optimization*, 4:309–326, 2004.

[Bix02] R. Bixby. Solving real-world linear programs: A decade and more of progress. *Operations research*, 50(1):3–15, 2002.

[Bix09] R. Bixby. Advanced Mixed Integer Programming: Solving MIPs in practice. In *Combinatorial Optimization at Work 2, Berlin*, 2009.

[BK73] C. Bron and J. Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *Communications of the ACM*, 16(9):575–577, 1973.

[BS08] E. Balas and A. Saxena. Optimizing over the split closure. *Mathematical Programming*, 113(2):219–240, 2008.

[BvdB04] H. Bodlaender and J. van den Broek. Treewidthlib: a benchmark for algorithms for treewidth and related graph problems. Technical report, Working paper, Department of Computer Science, Utrecht University, Utrecht, The Netherlands, 2004.

[Con13] S. Coniglio. Bound-optimal cutting planes. In *Proceedings of the 12th Cologne-Twente Worshop on Graphs and Combinatorial Optimization Workshop (CTW 2013), Enschede, The Netherlands*, pages 59–62, 2013.

[FG92] J. Forrest and D. Goldfarb. Steepest-edge simplex algorithms for linear programming. *Mathematical programming*, 57(1):341–374, 1992.

[FL07] M. Fischetti and A. Lodi. Optimizing over the first chvátal closure. *Mathematical Programming*, 110(1):3–20, 2007.

[GR77] D. Goldfarb and J. Reid. A practicable steepest-edge simplex algorithm. *Mathematical Programming*, 12(1):361–371, 1977.

[Har73] P. Harris. Pivot selection methods of the devex lp code. *Mathematical programming*, 5(1):1–28, 1973.

[JT96] D. Johnson and M. Trick. *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, Workshop, October 11-13, 1993*, volume 26. AMS Bookstore, 1996.

[Lüb10] M. Lübbecke. Column generation. *Wiley Encyclopedia of Operations Research and Management Science, John Wiley and Sons, Chichester, UK*, 2010.

[McC76] G. McCormick. Computability of global solutions to factorable nonconvex programs: Part iconvex underestimating problems. *Mathematical programming*, 10(1):147–175, 1976.

[NW88] G. Nemhauser and L. Wolsey. *Integer and combinatorial optimization*, volume 18. Wiley New York, 1988.

[SLL01] Jeremy G Siek, Lie-Quan Lee, and Andrew Lumsdaine. *Boost Graph Library: User Guide and Reference Manual, The.* Pearson Education, 2001.

[ZFB11] A. Zanette, M. Fischetti, and E. Balas. Lexicography and degeneracy: can a pure cutting plane algorithm work? *Mathematical programming*, 130(1):153–176, 2011.