Math. Program., Ser. A (2013) 141:165–192
DOI 10.1007/s10107-012-0513-3

FULL LENGTH PAPER

Strong lift-and-project cutting planes for the stable set problem

Monia Giandomenico · Fabrizio Rossi · Stefano Smriglio

Received: 1 February 2011 / Accepted: 7 January 2012 / Published online: 21 January 2012

Abstract A great deal of research has been focusing, since the early seventies, on finding strong relaxations for the stable set problem. Polyhedral combinatorics techniques have been at first developed to strengthen the natural linear formulation. Afterward, strong semidefinite programming relaxations have been deeply investigated. Nevertheless, the resulting integer programming (IP) algorithms cannot be regarded as being quite successful in practice, as most of the relaxations give rise to one out of two extreme situations: either provide weak bounds at low computational cost or give good bounds (sometimes excellent) but too demanding to compute. In this paper we present a method to bridge such a gap. In particular, a new lift-and-project relaxation is obtained by a problem-specific variant of the lifting operator $M(K, K)$ by Lovász and Schrijver, combined with Benders decomposition. This yields strong cutting planes, generated by solving a cut generating linear program. An extensive computational experience shows that embedding these cuts in a branch-and-cut framework significantly reduces the size of the enumeration trees as well as the CPU times with respect to state-of-the-art IP algorithms.

Keywords Stable set problem · Cutting planes · Lovász and Schrijver lift-and-project operators · Benders decomposition

Mathematics Subject Classification (2000) 90C10 (integer programming) · 90C57 (polyhedral combinatorics, branch-and-bound, branch-and-cut)

M. Giandomenico · F. Rossi · S. Smriglio (✉)
Dipartimento di Informatica, Università di L'Aquila, Via Vetoio, 67010 Coppito, (AQ), Italy
e-mail: stefano.smriglio@univaq.it

M. Giandomenico
e-mail: monia.giandomenico@univaq.it

F. Rossi
e-mail: fabrizio.rossi@univaq.it

1 Introduction

Let $G = (V, E)$ be an undirected graph with $|V| = n$ vertices. A vertex subset $S \subseteq V$ is called *stable* if no two elements of S are adjacent. The *stability number* $\alpha(G)$ of G is the size of a maximum cardinality stable set of G. The stable set problem (SSP) consists in computing a stable set of maximum cardinality, or, if a weight vector $w \in \mathbb{Q}^n_+$ is given, of maximum weight $\alpha_w(G)$. The SSP is NP-hard in the strong sense and hard even to approximate [17].

A natural 0–1 programming formulation for the SSP, known as *edge formulation*, is obtained by associating a binary variable x_i to each vertex i, taking value one if vertex i lies into the stable set and zero otherwise; and enforcing that, for each edge of G, at most one of its endpoints can belong to a stable set:

$$\begin{aligned} \max \quad & \sum_{i \in V} w_i x_i \\ \text{s.t.} \quad & \\ & x_i + x_j \leq 1 \quad (\forall \{i, j\} \in E) \qquad (1) \\ & x_i \in \{0, 1\} \quad (\forall i \in V) \end{aligned}$$

Inequalities (1) are called *edge* inequalities. The *stable set polytope*, $\text{STAB}(G) = \text{conv}\{\mathbf{x} \in \{0, 1\}^n : (1) \text{ hold } \forall \{i, j\} \in E\}$, is the convex hull of the incidence vectors of all stable sets of G. $\text{FRAC}(G) := \{x \in \mathbb{R}^n_+ : (1) \text{ hold } \forall \{i, j\} \in E\}$, denotes the polytope defined by the non negativity and edge inequalities.

Optimizing over FRAC(G) provides very weak upper bounds to $\alpha_w(G)$. A great effort has been devoted to strengthen the edge formulation by polyhedral combinatorics methods, which investigate the structure of STAB(G) in order to obtain valid inequalities. This kind of studies started in the early seventies, since Padberg [27] introduced the *clique* inequalities $\sum_{i \in C} x_i \leq 1$, for any vertex set $C \subseteq V$ inducing a *maximal* clique in G. On the practical side, the experience has been showing that clique inequalities are easy-to-manage cutting planes which do help close a relevant portion of the gap left by the edge inequalities. Even though the associate separation problem is strongly NP-hard, effective heuristics can be designed to solve it, representing mandatory ingredients of branch-and-cut algorithms for the SSP.

Padberg [27] also introduced the *odd hole* inequality $\sum_{i \in H} x_i \leq \left\lfloor \frac{|H|}{2} \right\rfloor$ for any vertex set H, $|H| \geq 5$, inducing a simple chordless cycle of odd cardinality (*odd hole*); and the *odd antihole* inequality $\sum_{i \in \bar{H}} x_i \leq 2$ for any vertex set $\bar{H}$ inducing an *odd antihole* (i.e., the complement of an odd hole). Odd hole inequalities were tested as cutting planes in [26] after being strengthened by sequential lifting, yielding some advantage on random graphs with up to 120 vertices with respect to clique inequalities. Several other families of valid inequalities have been investigated, such as *web* and *antiweb* [35], *wheel* inequalities [6] and *antiweb-wheel* inequalities [7]. Despite many nice theoretical results, none of them, to the best of our knowledge, gave any computational outcome.

The aforementioned inequalities are indeed special cases (or lifted versions) of *rank inequalities*, which have the form $\sum_{i \in W} x_i \leq \alpha(G[W])$, $W \subseteq V$. The overall computational contribution of general rank inequalities, regardless the structure of $G[W]$, has been evaluated in [31], where an effective *project-and-lift* (see [1] for a general description of such technique) separation heuristic was tested within a branch-and-cut algorithm. The performance of this algorithm turned out to get closer to that of pure combinatorial algorithms based on fast bounding techniques (valid only in the cardinality case) and smart enumeration schemes [24,25,32,34], which typically perform better. Nevertheless, not even general rank inequalities provide a conclusive contribution in closing the integrality gap, and large enumeration trees are in most cases unavoidable to certify optimality. This is also confirmed by a recent experience carried out by branch-and-cut algorithm including many separation routines [29].

Other methods have been applied to strengthen the natural formulation, which disdain the structure of $\mathrm{STAB}(G)$, namely, the *lift-and-project* methods developed by Balas et al. [2], Sherali and Adams [33], Lovász and Schrijver [22]. These first lift the initial formulation into a higher dimensional space where the formulation is strengthened, and then project it back onto the original space so as to obtain a relaxation contained in the initial one (see [8] for a comprehensive survey of these methods and references therein). The application of these procedures clearly highlighted the difficulty of the SSP, as the standard lift-and-project by Balas et al. and the Sherali-Adams procedures turned out to be impressively less effective than when applied to most 0–1 Programming problems.

Two lifting operators by Lovász and Schrijver [22] have been also investigated. The operator $\mathrm{M}_+(\cdot)$ has been applied to $\mathrm{FRAC}(G)$, yielding very strong upper bounds; unfortunately, it requires the lifted coefficient matrix to be positive semidefinite and the resulting semidefinite programming (SDP) problems create a very hard computational challenge. Finally, the operator $\mathrm{M}(K, K)$ has been applied to the polytope, denoted by $\mathrm{QSTAB}(G)$, defined by all clique and non-negativity inequalities. This yields a linear extended formulation which compares favourably to $\mathrm{M}_+(\mathrm{FRAC}(G))$ from both theoretical and computational perspectives and represents the starting point of this paper.

Overall, IP methods for the SSP suffer from the fact that most of the relaxations give rise to one out of two extreme situations: either they provide weak bounds (sometimes at low computational cost) or yield strong bounds but hardly available for practical implementations. In this paper we present a method to bridge such a gap. In particular, we introduce a relaxation of the extended formulation $\mathrm{M}(\mathrm{QSTAB}(G), \mathrm{QSTAB}(G))$ which preserves nice theoretical properties but, in addition, proves to be computationally much more tractable. Specifically, such a relaxation is projected onto the original space by the Benders decomposition. The resulting cutting planes, generated by solving a cut generating linear program (CGLP), are used in a branch-and-cut framework. An extensive computational experience is presented, showing that the cuts are very effective: upper bounds close (sometimes better) to those from SDP methods are obtained, yielding relevant reductions in the size of enumeration trees and CPU times.

The paper is organized as follows. In Sect. 2 a comparison among strong relaxations for the SSP is presented. In Sect. 3 the projection of M $(\mathrm{QSTAB}(G), \mathrm{QSTAB}(G))$ by the Benders decomposition is discussed. In Sect. 4 the new relaxation is introduced and, in Sect. 5, theoretical results are illustrated about its theoretical strength. In Sect. 6

the implementation issues are discussed and in Sect. 7 the computational experience is presented. Finally, in Sect. 8 some conclusions are drawn.

2 Survey of strong relaxations

In this section several known strong relaxations for the SSP are compared from both theoretical and computational perspectives. We start by summarizing SDP relaxations and then show how these compare to LP ones.

2.1 Semidefinite relaxations

A celebrated SDP relaxation for the SSP, known as *theta relaxation*, was introduced by Lovász in the seminal paper [21]. Let us consider the $n \times n$ matrix $X = xx^T$, where products $x_i x_j$ are replaced by quadratic variables x_{ij}. Lovász and Schrijver [22] introduce the augmented matrix Y

$$Y := \begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix}.$$

representing the product $\binom{1}{x}\binom{1}{x}^T$. Since Y is the product of a real matrix and its transpose, it is real, symmetric, square and positive semidefinite (psd). Then, an upper bound for the SSP is given by:

$$\begin{aligned} \max \; & \sum_{i=1}^{n} w_i Y_{0i} \\ \text{s.t.} \; & \\ & Y_{0i} = Y_{ii} \quad (i \in \{1, \dots, n\}) \\ & Y_{ij} = 0 \quad (i, j \in \{1, \dots, n\}, \{i, j\} \in E) \\ & Y \in S_+^{n+1} \end{aligned}$$

where S_+^{n+1} denotes the cone of real symmetric square psd matrices of order $n + 1$. This upper bound is denoted by $\theta(G, w)$ (or just $\theta(G)$ in the unweighted case). The projection TH(G) of the feasible region of this relaxation onto the subspace defined by the non-quadratic variables (convex, but not polyhedral in general) implies all clique inequalities. Formally, we have STAB$(G) \subseteq$ TH$(G) \subseteq$ QSTAB(G), with equality if and only if G is perfect [15]. In practice, $\theta(G)$ is much stronger than the bound obtained by using non-negativity and clique inequalities. Some classes of graphs for which this occurs are illustrated in Juhász [20].

It is worthwhile for our purposes to deepen this investigation by a computational insight. We implemented an effective cutting plane algorithm based on a heuristic separation of clique inequalities (see Sect. 6 for the details). The upper bound computed by such an algorithm is, from now on, denoted by UB_{clq}. In Table 1 this is compared with $\theta(G)$ for all the DIMACS challenge benchmark instances [19] (see Sect. 7), with

Table 1 DIMACS graphs: clique inequalities versus $\theta(G)$

Graph	$\|V\|$	$\|E\|$	$\alpha(G)$	UB_{clq}	$\theta(G)$	$\frac{UB_{\text{clq}}-\theta}{UB_{\text{clq}}-\alpha}\%$
brock200_1	200	5,066	21	38.06	27.50	61.90
brock200_2	200	10,024	12	21.33	14.22	76.21
brock200_3	200	7,852	15	27.34	18.82	69.04
brock200_4	200	6,811	17	30.67	21.29	68.62
C.125.9	125	787	34	43.06	37.89	57.06
C.250.9	250	3,141	44	71.38	56.24	55.30
c-fat200-1	200	18,336	12	12.00	12.00	–
c-fat200-2	200	16,665	24	24.00	24.00	–
c-fat200-5	200	11,427	58	66.67	60.34	73.01
DSJC125.1	125	736	34	43.15	38.39	52.02
DSJC125.5	125	3,891	10	15.44	11.47	72.98
DSJC125.9	125	6,961	4	4.69	4.00	100.00
mann_a9	45	72	16	18.50	17.47	41.20
mann_a27	378	702	126	135.00	132.76	24.89
gen200_p0.9_44	200	1,990	44	44.00	44.00	–
gen200_p0.9_55	200	1,990	55	55.00	55.00	–
hamming6-2	64	192	32	32.00	32.00	–
hamming6-4	64	1,312	4	5.33	5.33	0.00
hamming8-2	256	1,024	128	128.00	128.00	–
hamming8-4	256	11,776	16	16.00	16.00	–
johnson8-2-4	28	168	4	4.00	4.00	–
johnson8-4-4	70	560	14	14.00	14.00	–
johnson16-2-4	120	1,620	8	8.00	8.00	–
keller4	171	5,100	11	14.82	14.01	21.20
p_hat300_1	300	33,917	8	15.30	10.10	71.23
p_hat300_2	300	22,922	25	33.59	27.00	76.72
p_hat300_3	300	11,460	36	54.36	41.16	71.90
san200_0.7-1	200	5,970	30	30.00	30.00	–
san200_0.7-2	200	5,970	18	19.04	18.00	100.00
san200_0.9-1	200	1,990	70	70.00	70.00	–
san200_0.9-2	200	1,990	60	60.00	60.00	–
san200_0.9-3	200	1,990	44	44.00	44.00	–
sanr200_0.7	200	6,032	18	33.39	23.80	62.31
sanr200_0.9	200	2,037	42	59.82	49.30	59.03

less than 400 vertices. In 14 out of 34 instances the clique inequalities completely close the integrality gap. In one of the remaining 20 instances (`hamming6-4`) UB_{clq} equals $\theta(G)$; in three graphs, namely, `mann_a9`, `mann_a27` and `keller4` the percentage gap closed by $\theta(G)$ w.r.t. UB_{clq} (last column) is less than 50%; in 14 cases it ranges within [50%, 80%] and in two cases $\theta(G)$ completely closes the gap left by the clique inequalities.

The results give evidence of the advantage of the theta relaxation with respect to the linear relaxation based on clique inequalities (i.e., a reference relaxation for traditional methods from polyhedral combinatorics). Thanks to recent progress in dedicated SDP solvers [28,23], implementing branch-and-bound algorithms based on the theta bound looks viable, although not straightforward. One such algorithm has been recently tested in [37], showing a quite promising behaviour. A new relaxation, having the form of an ellipsoid, has been also recently proposed in [13], providing an upper bound equal to $\theta(G)$.

Even stronger relaxations than $\mathrm{TH}(G)$ have been considered. In Burer and Vandenbussche [5] the application of the Lovász-Schrijver $\mathrm{M}_+(\cdot)$ operator to $\mathrm{FRAC}(G)$ is investigated. Indeed, $\mathrm{M}_+(\mathrm{FRAC}(G))$ can be seen as being obtained by adding some linear inequalities to the Lovász θ relaxation (see [12] for a detailed description). $\mathrm{N}_+(\mathrm{FRAC}(G))$, i.e., the projection of $\mathrm{M}_+(\mathrm{FRAC}(G))$ onto the non-quadratic space, has stronger theoretical properties than $\mathrm{TH}(G)$. In particular, Lovász and Schrijver [22] showed that $\mathrm{N}_+(\mathrm{FRAC}(G))$ satisfies all clique, odd cycle, odd antihole and odd wheel inequalities. Giandomenico and Letchford [11] showed that it also satisfies all web inequalities.

On the computational side, $\mathrm{M}_+(\mathrm{FRAC}(G))$ represents a very tough computational challenge. Burer and Vandenbussche [5] report experiments carried out by a tailored lagrangian method, showing that the upper bounds can be very strong. Unfortunately, in many instances of the DIMACS challenge benchmark library (with up to 300 vertices), the best solution value found by their algorithm is higher than $\theta(G)$, and often takes long CPU times to be computed.

Other related relaxations, still dominating the Lovász theta relaxation, have been investigated by Gruber and Rendl [16] and Dukanovic and Rendl [9] (see [12] for further details). Again, the upper bounds obtained were very good but computationally very hard to compute.

In summary, although SDP problems can be solved (to arbitrary precision) in polynomial time, optimizing over SDP relaxations dominating the Lovász theta relaxation turned out to be much harder than computing $\theta(G)$. Therefore, such valuable upper bounds are hardly accessible in a branch-and-bound scheme. This motivates the investigation of strong linear relaxations, addressed in the next subsections.

2.2 SDP versus LP relaxations

Another insightful experiment by Burer and Vandenbussche [5] compare the bound given by $\mathrm{M}_+(\mathrm{FRAC}(G))$ with the one of the Sherali-Adams (linear) relaxation $\mathrm{M}(\mathrm{FRAC}(G))$. Their results are reported in Table 2, in which we also included UB_{clq}.

The table highlights that the Sherali-Adams relaxation is impressively weaker than $\mathrm{M}_+(\mathrm{FRAC}(G))$, where matrix Y is imposed to be positive semidefinite. Notice that $\mathrm{M}(\mathrm{FRAC}(G))$ is even much worse than UB_{clq}.

Another insightful comparison between SDP and LP relaxations is illustrated in [3]. The authors compare the lift-and-project relaxation of Balas, Ceria and Cornuéjols [2] (which is dominated by $\mathrm{M}(\mathrm{FRAC}(G))$) with two stronger procedures in which the

Table 2 $M_+(\text{FRAC}(G))$ versus $M(\text{FRAC}(G))$

Graph	$\alpha(G)$	UB_{clq}	$M(FRAC(G))$	$M_+(FRAC(G))$
brock200_1	21	38.06	66.66	27.98
brock200_2	12	21.33	66.66	17.08
brock200_3	15	27.34	66.66	20.79
brock200_4	17	30.67	66.66	22.84
c-fat200-5	58	66.67	66.66	58.17
mann_a9	16	18.50	18.00	17.17
hamming6-4	4	5.33	21.33	4.54
keller4	11	14.82	57.00	15.41
p_hat300_1	8	15.30	100.00	18.66
p_hat300_2	25	33.59	100.00	30.1
p_hat300_3	36	54.36	100.00	43.32
san200_0.7-2	18	19.04	66.66	20.01
sanr200_07	18	33.39	66.66	24.97
sanr200_09	42	59.82	66.66	49.31

disjunctions based on clique inequalities (in complemented version, as they address the maximum clique problem) replace simple disjunctions. In addition, the condition $Y \succeq 0$ is replaced by an infinite family of linear inequalities, namely, the *positive semidefinite (psd) constraints*. This paper represents the only attempt to embed strong (unstructured) cutting planes within a branch-and-cut algorithm for the SSP. Two relevant indications come out of this study. First, although the psd constraints can be separated in polynomial time, they are not easily manageable as cutting planes (after being projected onto the linear space) and they do not look promising in practice. Second, the experimental indications highlights that clique inequalities may play an important role. This direction has been extensively investigated in [12], as described in the next subsection.

2.3 The M(QSTAB(G), QSTAB(G)) relaxation

The $M(K, K)$ operator, introduced in Lovász-Schrijver [22], works as follows. For any pair of linear inequalities $\alpha x - \beta \geq 0$ and $\alpha' x - \beta' \geq 0$, of the initial relaxation K, the 'product' inequality $\left(-\beta \; \alpha^T\right) Y \begin{pmatrix} -\beta' \\ \alpha' \end{pmatrix} \geq 0$ is computed. The products $x_i x_j$, for all $1 \leq i < j \leq n$, are then replaced with new variables x_{ij}, and the terms x_i^2, for $1 \leq i \leq n$, are replaced with x_i (which is valid when x_i is binary.) This yields an extended LP formulation which is provably stronger than the original.

In [12] the $M(K, K)$ operator has been applied to the polytope QSTAB(G):

$$\sum_{i \in C} x_i \leq 1 \quad (C \in \Omega) \tag{2}$$

$$x_i \geq 0 \quad (i \in V) \tag{3}$$

where Ω denotes the set of all maximal cliques of G. This gives the relaxation M(QSTAB(G),QSTAB(G)) of the following form:

$$\max \sum_{i \in V} x_i$$

$$\text{s.t.}$$

$$\sum_{i \in C} x_i \le 1 \qquad (C \in \Omega) \tag{4}$$

$$\sum_{i \in C \cup C'} x_i - \sum_{\{i,j\} \in \bar{E}(C:C')} x_{ij} \le 1 \quad (C, C' \in \Omega, C \ne C') \tag{5}$$

$$-x_i + \sum_{j \in C : \{i,j\} \in \bar{E}} x_{ij} \le 0 \quad (C \in \Omega, i \in V \backslash C) \tag{6}$$

$$x_{ij} = 0 \qquad (\{i,j\} \in E)$$

$$x_{ij} \ge 0 \qquad (\{i,j\} \in \bar{E})$$

$$x_i \ge 0 \qquad (i \in V) \tag{7}$$

where $\bar{E} := \{\{i,j\} \subset V : \{i,j\} \notin E\}$ denotes the set of 'non-edges', and $\bar{E}(C : C') = \left\{\{i,j\} \in \bar{E} : i \in C, j \in C'\right\}$.

Inequalities (4) (5) are obtained by multiplying two clique inequalities ((4) corresponding to $C = C'$) and are referred to as *clique-product* inequalities (CPIs). Inequalities (6) (7) are obtained by multiplying a clique with a non-negativity inequality ((7) corresponding to $i \in C$) and are referred to as *clique-variable* inequalities (CVIs). Therefore, M(QSTAB(G), QSTAB(G)) contains $|\Omega|+|\Omega|(|\Omega|-1)/2+n|\Omega|$ inequalities, and is non-compact, as $|\Omega|$ is exponential in n in general.

For the sake of brevity, we refer to M(QSTAB(G), QSTAB(G)) simply as M(K, K) and let N(K, K) denote its projection onto the subspace of the original (non-quadratic) variables. N(K, K) is stronger than the Sherali-Adams relaxation. Precisely, the following inclusions hold [22]:

$$\text{N}(K, K) \subseteq \text{N}(\text{QSTAB}(G)) \subseteq \text{N}(\text{FRAC}(G)).$$

Moreover, N(K, K) neither contains nor is contained in TH(G) or N_+(FRAC(G)). However, it was proved in [12] that N(K, K) implies all web and antiweb inequalities (and therefore all edge, clique, odd hole and odd antihole inequalities), together with various lifted versions. A summary of theoretical results, showing how N(K, K) compares favourably to strong SDP relaxations, is illustrated in Table 3. The relaxations $\text{N}_j(K, K)$ and $\cap_{j \in V} \text{N}_j(K, K)$ reported in the last two lines of the table will be defined in Sect. 4.

These encouraging theoretical indications have an experimental confirmation. An extensive computational experience [12] showed that, in several cases, M(K, K) provides upper bounds which are smaller than $\theta(G)$. This valuable result still does not lead to a practical branch-and-cut algorithm, since M(K, K) is an extended non-compact formulation showing also huge degeneracy. As a first step towards a practical

Table 3 Summary of theoretical results

Relaxation	Implied inequalities
TH(G)	Clique [15]
$\mathrm{N}_+(FRAC(G))$	Clique, odd-cycle, odd-antihole, odd-wheel [22] and web [11]
$\mathrm{N}(K, K)$	Clique, web, antiweb ($\Rightarrow$ odd-hole and odd-antihole) and their lifted versions [12]
$\mathrm{N}_j(K, K)$	Clique
$\cap_{j\in V}\mathrm{N}_j(K, K)$	Clique, antiweb ($\Rightarrow$ odd-hole and odd-antihole), a subclass of web and their lifted versions

implementation, in the next section the projection of $\mathrm{M}(K, K)$ onto the linear space by the Benders decomposition is illustrated.

3 Projection

Representing variables x_{ij} by the vector y, the $\mathrm{M}(K, K)$ relaxation can be rewritten in the following compact form:

$$\begin{aligned} \max\ & \mathbf{1}^T x + \mathbf{0}^T y \\ \text{s.t.}\ & \\ & Ax \le \mathbf{1} \\ & Bx + Dy \le d \\ & x \in \mathbb{R}^n_+,\ y \in \mathbb{R}^q_+ \end{aligned} \tag{8}$$

where $A \in \{0, 1\}^{m\times n}$ is the coefficient matrix corresponding to the clique inequalities, $B \in \{-1, 0, 1\}^{p\times n}$ and $D \in \{-1, 0, 1\}^{p\times q}$ the matrices corresponding to CVIs and the remaining CPIs, and $d \in \{0, 1\}^p$. One way to obtain the relaxation $\mathrm{N}(K, K)$ is to apply the Benders reformulation [4] to $\mathrm{M}(K, K)$. This yields the following form for $\mathrm{N}(K, K)$:

$$\max \quad \mathbf{1}^T x + \eta$$

s. t.

$$Ax \le \mathbf{1} \tag{9}$$

$$u^T(d - Bx) \ge \eta \quad (u \in \mathrm{Ext}(Q)) \tag{10}$$

$$v^T(d - Bx) \ge 0 \quad (v \in \mathrm{Ray}(Q)) \tag{11}$$

$$x \in \mathbb{R}^n_+,\ \eta \in \mathbb{R}$$

where sets $\mathrm{Ext}(Q)$ and $\mathrm{Ray}(Q)$ contain respectively the extreme points and extreme rays of the polyhedron $Q := \{v \in \mathbb{R}^p : v^T D \ge \eta, v \ge \mathbf{0}\}$. Here, η represents the contribution of the y variables to the objective value. In our case, $\eta = 0$. Therefore,

$Q = \{v \in \mathbb{R}^p : v^T D \geq 0, v \geq \mathbf{0}\}$ is a polyhedral cone and $\text{Ext}(Q)$ contains only the zero vector. As a consequence, constraints (10) disappear from the formulation.

Since $\text{Ray}(Q)$ has an exponential size, a cutting plane algorithm is needed to optimize over $\text{N}(K, K)$. In particular, constraints (11), often referred to as *feasibility* cuts, have to be dynamically generated. This can be done by solving a linear program, as described in [10]. If x^* denotes the (fractional) point to be separated, the *cut generating linear program* (CGLP) reads as:

$$\begin{gathered} \min v^T (d - Bx^*) \\ \text{s.t.} \\ v^T D \geq \mathbf{0} \\ \mathbf{1}^T v = 1 \\ v \geq \mathbf{0} \end{gathered} \tag{12}$$

where the objective function measures the cut violation and the constraint (12), referred to as *normalization* constraint, is used to truncate the cone Q. Indeed, the extreme rays of Q are in a one-to-one correspondence to the vertices of the polyhedron $\{v \in \mathbb{R}^p : v^T D \geq 0, \mathbf{1}^T v = 1, v \geq \mathbf{0}\}$.

If we denote by $v_{CC'}$ the dual variables associated to inequalities (5), and by u_{C_i} the dual variables associated to inequalities (6), the CGLP takes the form:

$$\begin{aligned} \min \sum_{C \in \Omega, i \in V \setminus C} x_i^* u_{C_i} + \sum_{C, C' \in \Omega} \left(1 - \sum_{i \in C \cup C'} x_i^*\right) v_{CC'} \\ \text{s.t.} \\ \sum_{C \in \Omega : j \in C} u_{C_i} + \sum_{C \in \Omega : i \in C} u_{C_j} - \sum_{C, C' \in \Omega : i \in C, j \in C'} v_{CC'} \geq 0 \quad (\forall \{i, j\} \in \bar{E}) \\ \sum_{C \in \Omega, i \in V \setminus C} u_{C_i} + \sum_{C, C' \in \Omega} v_{CC'} = 1 \\ u_{C_i}, v_{CC'} \geq 0 \quad (C, C' \in \Omega, C \neq C', i \in V \setminus C) \end{aligned}$$

A preliminary experience showed that a cutting plane algorithm based on this CGLP can reach upper bounds close to those computed in [12] by optimizing over $\text{M}(K, K)$ (the bound impairment due to the projection does not exceed 10%) for sparse (up to 15% density) and small graphs (up to 150 vertices). However, generating these cuts turned out to be too computationally expensive for denser and larger graphs. In fact, although the number of rows of the CGLP decreases, the number of columns, that roughly depends on the square of $|\Omega|$, considerably increases with graph density and size. In addition, Benders cuts alternate between sparse cuts with "nice" coefficients and dense cuts with large coefficients dynamism. The latter tend to interfere with the branching process and worsen the branch-and-cut performance, even if they are effective in closing the integrality gap.

In summary, the method has the great merit of providing strong bounds by a linear formulation in the space of the original variables. However, some computational difficulties still limit its applicability to several graphs of interest. In the next section, we introduce a new relaxation able to preserve the quality of the bounds but showing a remarkably easier computational tractability.

4 The $\mathrm{N}_j(K, K)$ relaxation

The structure of the SSP can be exploited so as to obtain a more compact relaxation, while preserving the strength of $\mathrm{M}(K, K)$.

For each $j \in V$, we denote by $V(j)$ the neighborhood of j, that is, the set of vertices adjacent to j; let also $\bar{V}(j) = V \setminus V(j)$. By extension, we let $V(S) = \cup_{j \in S} V(j)$, for a given vertex subset S. Let also $\Omega(j)$ be the set of all maximal cliques containing j and $\bar{\Omega}(j) = \Omega \setminus \Omega(j)$. Note that each clique in $\bar{\Omega}(j)$ contains at least one vertex in $\bar{V}(j)$.

The idea is to consider, for a given vertex $j \in V$, only CPIs containing variables x_{jk}, for $\{j, k\} \in \bar{E}$. These CPIs are in fact those associated to clique pairs C, C' in which $C \in \Omega(j)$ and $C' \in \bar{\Omega}(j)$ and may contain variables x_{hk}, for $\{h, k\} \in \bar{E}$ and $h, k \neq j$. The relaxation then includes only CVIs containing such variables. Formally, we define the relaxation $\mathrm{M}_j(K, K)$ as follows:

$$\max \sum_{i \in V} x_i$$

$$\text{s.t.}$$

$$\sum_{i \in C \cup C'} x_i - \sum_{\{i,k\} \in \bar{E}(C:C')} x_{ik} \leq 1 \qquad (C \in \Omega(j), C' \in \bar{\Omega}(j)) \tag{13}$$

$$-x_i + \sum_{k \in C:\{i,k\} \in \bar{E}} x_{ik} \leq 0 \qquad (C \in \bar{\Omega}(j), i \in \{j\} \cup V(j)) \tag{14}$$

$$-x_i + \sum_{k \in C:\{i,k\} \in \bar{E}} x_{ik} \leq 0 \quad (C \in \Omega(j), i \in \bar{V}(j) \cup V(\bar{V}(j))) \tag{15}$$

$$x_{ik} = 0 \qquad (\{i, k\} \in E)$$

$$x_{ik} \geq 0 \qquad (\{i, k\} \in \bar{E})$$

$$x_i \geq 0 \qquad (i \in V) \tag{16}$$

Observe that the CVIs associated with a clique-vertex pair (C, i) such that $C \in \bar{\Omega}(j)$ and $i \in \bar{V}(j)$, or $C \in \Omega(j)$ and $i \in V(j)$ but $i \notin C'$, for any $C' \in \bar{\Omega}(j)$, are discarded. Moreover, variables x_{ik}, $\{i, k\} \in \bar{E}$ such that $i, k \in \bar{V}(j)$ or $i, k \in V(j)$ but $i, k \notin C'$, for any $C' \in \bar{\Omega}(j)$, do not appear in $\mathrm{M}_j(K, K)$.

Example 1 Consider the graph in Fig. 1 and let $j = 1$; we have $\Omega(j) = \{\{1, 2\}, \{1, 7\}\}$, $\bar{\Omega}(j) = \{\{2, 3\}, \{3, 4\}, \{4, 5\}, \{5, 6\}, \{6, 7\}\}$. A CVI of the form (14) is generated by $C = \{4, 5\} \in \bar{\Omega}(1)$ and $i = 2 \in V(1)$; a CVI of the form (15) by $C = \{1, 7\} \in \Omega(1)$ and $i = 3 \in \bar{V}(1)$ or $C = \{1, 7\} \in \Omega(1)$ and $i = 2 \in V(\bar{V}(1))$. An example of a discarded CVI is $C = \{4, 5\}$ and $i = 3$.

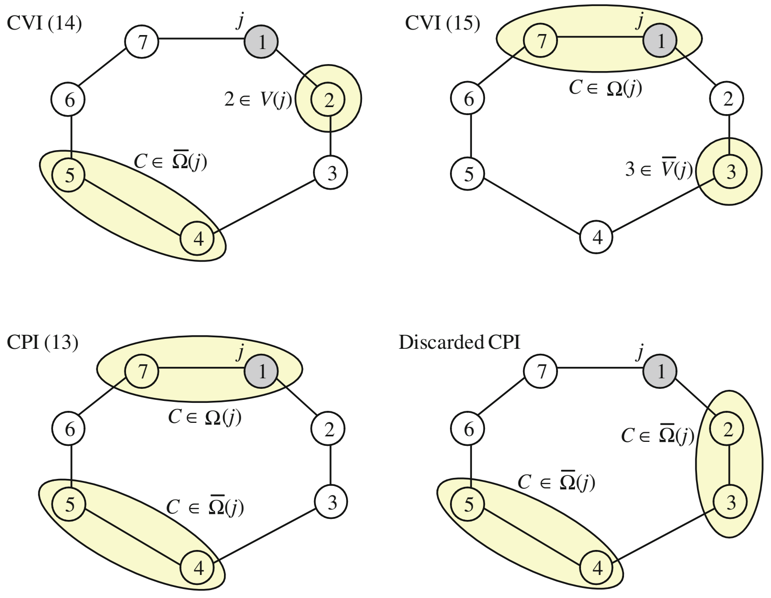

Fig. 1 Example

A CPI of the form (13) is generated by $C = \{1, 7\} \in \Omega(1)$ and $C' = \{4, 5\} \in \bar{\Omega}(1)$. An example of a discarded CPI is given by $C = \{1, 7\}$, $C' = \{1, 2\}$ and $C = \{2, 3\}$, $C' = \{4, 5\}$. Overall, twelve CVIs out of thirty-five (non trivial) and eleven CPIs out of twenty-one (non-clique) are discarded. Finally, three of the fourteen quadratic variables (namely, x_{35}, x_{36} and x_{46}) are dropped.

The projection of $\mathrm{M}_j(K, K)$ onto the space of the original variables is denoted by $\mathrm{N}_j(K, K)$. This has the following nice property:

Theorem 1 $\mathrm{N}_j(K, K) \subseteq \mathrm{QSTAB}(G)$, $\forall j \in V$.

Proof We show that all maximal clique inequalities can be obtained by conic combination of CVIs (14) and CPIs (13). Let $Q \in \Omega(j)$, the corresponding clique inequality is obtained by choosing a generic clique $Q' \in \bar{\Omega}(j)$ and summing up the (single) CPI (13) where $C = Q$ and $C' = Q'$, together with all the CVIs (15) where $C = Q$ and $i \in Q' \backslash Q$. Similarly, let $Q \in \bar{\Omega}(j)$, then the corresponding clique inequality is obtained by choosing a generic clique $Q' \in \Omega(j)$ and summing up the CPI (13) where $C' = Q$ and $C = Q'$, together with all the CVIs (14) where $C = Q$, and $i \in Q' \backslash Q$. □

By construction, we have $\cap_{j \in V} \mathrm{M}_j(K, K) = \mathrm{M}(K, K)$. When the relaxations are projected this is no longer true, that is, $\mathrm{N}(K, K) \subseteq \cap_{j \in V} \mathrm{N}_j(K, K)$. In the next section we show that most of the theoretical results proved in [12] for $\mathrm{N}(K, K)$ still hold for $\cap_{j \in V} \mathrm{N}_j(K, K)$. But the great advantage of $\mathrm{M}_j(K, K)$ is on the practical side. The projection technique described in Sect. 3 applied to $\mathrm{M}_j(K, K)$ results in

CGLPs impressively more tractable than those obtained from the $\mathrm{M}(K, K)$ relaxation. Indeed, the actual reduction in CGLPs size is crucial to develop a cost-effective implementation, as shown in Sect. 7.

5 On the strength of the closure $\cap_{j\in V} \mathrm{N}_j(K, K)$

Let p and q be integers such that $q \geq 2$ and $p > 2q + 1$. Here arithmetic modulo p is used. A $(p, q)-$web, denoted by $W(p, q)$, is a graph with vertices $\{1, \ldots, p\}$ and edges from i to $i + q, \ldots, i + p - q$, for every $i = 1, \ldots, p$. The web inequality takes the form $\sum_{i=1,\ldots,p} x_i \leq q$. A $(p, q)-$antiweb, denoted by $AW(p, q)$, is the complement graph of the web $W(p, q)$. The antiweb inequality takes the form $\sum_{i=1,\ldots,p} x_i \leq \lfloor p/q \rfloor$. Web and antiweb inequalities have been introduced in [35]. Note that they have odd hole and antihole inequalities as special cases. In what follows we denote by $r = p - \lfloor \frac{p}{q} \rfloor \cdot q$ the reminder in the Euclidean division of p by q and by $\omega = \lfloor p/q \rfloor$ the cardinality of a maximum clique in a web. We will also use the fact that inequalities (13), (14) and (15), are implied by $\mathrm{M}_j(K, K)$ even when C and C' are not maximal cliques.

Theorem 2 $\mathrm{N}_j(K, K)$ *satisfies all antiweb inequalities corresponding to antiwebs containing vertex* j.

Proof Let $AW(p, q)$ be an antiweb containing vertex j. Because of the symmetry, without loss of generality, we can suppose $j = 1$. Let us consider:
$\lfloor \frac{p}{q} \rfloor$ CPIs (13) defined by the following clique pairs

- $C = \{1, \ldots, q\}$, $C' = \{hq + 1, hq + 2, \ldots, (h + 1)q\}$, $\forall h = 1, \ldots, \lfloor \frac{p}{q} \rfloor - 1$;
- $C = \{1, \ldots, q\}$, $C' = \{p - q + 1, p - q + 2, \ldots, p\}$;

$q(\lfloor \frac{p}{q} \rfloor - 1)$ CVIs (14) where

- $i \in \{1, \ldots, q\}$ and $C = \{i + hq, i + hq + 1, \ldots, i + (h + 1)q - 1\}$, $\forall h = 1, \ldots, \lfloor \frac{p}{q} \rfloor - 1$

and $(q - r)$ CVIs (15) where

- $C = \{1, 2, \ldots, q\}$ and $i \in \{p - q + 1, p - q + 2, \ldots, p - r\}$.

Summing all the above CPIs we obtain:

$$\sum_{i=1,\ldots,p} x_i + (\lfloor p/q \rfloor - 1) \sum_{i=1,\ldots,q} x_i + \sum_{i=p-q+1,\ldots,p-r} x_i +$$
$$- \sum_{i=1,\ldots,q;\ k=i+q,\ldots,i+p-q} x_{ik} - \sum_{i=1,\ldots,q;\ k=p-q+1,\ldots,\bar{k}} x_{ik} \leq \lfloor p/q \rfloor$$

where $\bar{k} = \min\{p - r, p + i - q\}$.

Summing all the above CVIs we obtain:

$$-(\lfloor p/q \rfloor - 1) \sum_{i=1,\dots,q} x_i - \sum_{i=p-q+1,\dots,p-r} x_i + \sum_{i=1,\dots,q;\ k=i+q,\dots,i+p-q} x_{ik}$$
$$+ \sum_{i=1,\dots,q;\ k=p-q+1,\dots,\bar{k}} x_{ik} \leq 0$$

Finally, summing the last two inequalities the antiweb inequality $\sum_{i=1,\dots,p} x_i \leq \lfloor p/q \rfloor$ is obtained. □

Theorem 3 $\mathrm{N}_j(K, K)$ *satisfies all web inequalities corresponding to webs containing vertex j and such that $r \leq \omega$.*

Proof Let $W(p, q)$ be a web, with $r \leq \omega$, containing vertex j. Again, we can suppose $j = 1$. If $r = 0$, the web inequality is nothing but the sum of q clique inequalities corresponding to the maximal cliques $C = \{i + hq, h = 0, \dots, \omega - 1\}$, for all $i = 1, \dots, q$. Therefore, in what follows, we consider $r > 0$. Let us consider q CPIs (13) defined by the following pairs of cliques

- $C = \{1 + hq : h = 0, \dots, \omega - r\} \cup \{2 + hq : h = \omega - r + 1, \dots, \omega - 1\}$ and $C' = \{i + hq : h = 0, \dots, \omega - r\} \cup \{i + hq + 1 : h = \omega - r + 1, \dots, \omega - 1\}$ for all $i = 2, \dots, q$;
- $C = \{1 + hq : h = 0, \dots, \omega - r\} \cup \{2 + hq : h = \omega - r + 1, \dots, \omega - 1\}$ and $C' = \{p - h(q + 1) : h = 0, \dots, r - 1\}$;

here, the assumption $r \leq \omega$ guarantees C' is a clique. Note that in the last CPI, C' may not be maximal. Next we consider $\omega(q - 1)$ CVIs (14) where

- $i \in \{1 + hq : h = 0, \dots, \omega - r\} \cup \{2 + hq : h = \omega - r + 1, \dots, \omega - 1\}$ and $C = \{i + h, i - q + h\}$ for all $h = 1, \dots, q - 1$.

here the clique C is always an edge. Summing all the above CPIs we obtain:

$$\sum_{i=1,\dots,p} x_i + (q-1) \sum_{h=0,\dots,\omega-r} x_{1+hq} + (q-1) \sum_{h=\omega-r+1,\dots,\omega-1} x_{2+hq} +$$
$$- \sum_{h=0,\dots,\omega-r;\ k=2+hq,\dots,h(q+1)} x_{1+hq\ k} - \sum_{h=0,\dots,\omega-r;\ k=hq,\dots,2+h(q-1)} x_{1+hq\ k} +$$
$$- \sum_{h=\omega-r+1,\dots,\omega-1;\ k=3+hq,\dots,1+h(q+1)} x_{2+hq\ k} +$$
$$- \sum_{h=\omega-r+1,\dots,\omega-1;\ k=1+hq,\dots,3+h(q-1)} x_{2+hq\ k} \leq q$$

Summing all the above CVIs we obtain:

$$\begin{aligned}
&-(q-1)\sum_{h=0,\dots,\omega-r} x_{1+hq} - (q-1)\sum_{h=\omega-r+1,\dots,\omega-1} x_{2+hq}\\
&+\sum_{h=0,\dots,\omega-r;\ k=2+hq,\dots,h(q+1)} x_{1+hq\ k} + \sum_{h=0,\dots,\omega-r;\ k=hq,\dots,2+h(q-1)} x_{1+hq\ k}\\
&+\sum_{h=\omega-r+1,\dots,\omega-1;\ k=3+hq,\dots,1+h(q+1)} x_{2+hq\ k}\\
&+\sum_{h=\omega-r+1,\dots,\omega-1;\ k=1+hq,\dots,3+h(q-1)} x_{2+hq\ k} \leq 0
\end{aligned}$$

Finally, summing the last two inequalities the web inequality $\sum_{i=1,\dots,p} x_i \leq q$ is obtained. □

This theorem gives only a sufficient condition, that is, there exist web inequalities (containing vertex j) which are implied by $\mathrm{N}_j(K, K)$ but do not satisfy $r \leq \omega$. Indeed, one can prove that not all such web inequalities are implied by $\mathrm{N}_j(K, K)$.

Overall, Theorems 2 and 3 imply the following

Corollary 1 $\cap_{j\in V} \mathrm{N}_j(K, K)$ *satisfies all clique, antiweb (and therefore all odd hole and antihole) and a subclass of web inequalities which includes those with* $r \leq \omega$.

This differs from what was proved in [12] for $\mathrm{N}(K, K)$ only because it implies all the web inequalities as well. However, the class of web inequalities implied by $\cap_{j\in V} \mathrm{N}_j(K, K)$ is very large. It includes all inequalities associated to $(p, q)-$webs with $p \geq (q-1)(q+1)$ in addition to those with $r = 0, 1, 2$. In particular, it includes all web inequalities with $q = 3$ and $q = 4$.

5.1 Sequential lifting

We now deal with inequalities obtained by the lifting procedure described in [12], which works as follows. Let $a^T x \leq b$ be a valid inequality for STAB(G), and let S be a stable set in G. Let us look at the new graph $\tilde{G} = (\tilde{V}, \tilde{E})$ obtained from G by adding an extra vertex w adjacent to every vertex in $S \cup V(S)$. A valid inequality $\tilde{a}^T x \leq b$ for STAB($\tilde{G}$), obtained by *lifting on* S, is given by $\tilde{a}_i = a_i$ for all $i \in V$ and $\tilde{a}_w = \sum_{i\in S} a_i$.

If a valid inequality for STAB(G) is implied by $\mathrm{N}(K, K)$, then any inequality obtained by lifting on a given stable set S is also valid [12]. In this subsection, we prove that the same result holds for $\mathrm{N}_j(K, K)$.

Theorem 4 *Let* $a^T x \leq b$ *be implied by* $\mathrm{N}_j(K, K)$*, and* S *be a stable set in* G*. The inequality* $\tilde{a}^T x \leq b$*, obtained by lifting on* S*, is also implied by* $\mathrm{N}_j(K, K)$*.*

Proof Assume that the inequality $a^T x \leq b$ can be expressed by a conic combination of: (i) a family R of CVIs of the form (14) (possibly (16) if $i \in C$) associated to the clique-vertex pairs $\{(C_r, i_r) : r \in R\}$; (ii) a family Z of CVIs of the form (15) (possibly (16) if $i \in C$) associated to the clique-vertex pairs $\{(C_z, i_z) : z \in Z\}$; and (iii) a family T of CPIs of the form (13) associated to clique-clique pairs $\{(C_t, C'_t) : t \in T\}$.

For a given vertex i, let R_i (Z_i) and T_i be the sets of CVIs in R (in Z) and CPIs in T in which the variable x_i has a non zero coefficient. Analogously, for a given pair of vertices $\{i, k\} \in \bar{E}$, let R_{ik} (Z_{ik}) and T_{ik} be the sets of CVIs in R (Z) and CPIs in T in which the variable x_{ik} has a non zero coefficient.

Let $\lambda_r > 0, \lambda_z > 0$ and $\lambda_t > 0$, for $r \in R, z \in Z$ and $t \in T$, be the multipliers given to the corresponding CVIs and CPIs in the combination. The following equalities hold

$$-\sum_{r \in R_i} \lambda_r - \sum_{z \in Z_i} \lambda_z + \sum_{t \in T_i} \lambda_t = a_i \qquad (\forall i \in V). \tag{17}$$

$$\sum_{r \in R_{ik}} \lambda_r + \sum_{z \in Z_{ik}} \lambda_z - \sum_{t \in T_{ik}} \lambda_t = 0 \qquad (\forall \{i, k\} \in \bar{E}). \tag{18}$$

We now show that the lifted inequality $\tilde{a}^T x \leq b$ can be obtained by combining some new CVIs with the CVIs in R, Z and CPIs in T after (eventually) being modified to include vertex w. We use the notation $\bar{E}^+ := \{\{i, k\} \subseteq \tilde{V} : \{i, k\} \notin \tilde{E}\}$.

For each $r \in R$ such that $i_r \in S$, the new CVI

$$-x_w + \sum_{k \in C_r : \{w,k\} \in \bar{E}^+} x_{wk} \leq 0 \tag{19}$$

associated to vertex w and clique C_r, is added to the conic combination with multiplier λ_r. This is still of the form (14) or (16). To explain this, recall that w is adjacent to all vertices in $S \cup V(S)$, $i_r \in S$ and $i_r \in \{j\} \cup V(j)$; therefore $w \in V(j)$. Note that w could be adjacent to all vertices in C_r and the inequality (19) reduces to $-x_w \leq 0$. The same construction is applied for each $z \in Z$ such that $i_z \in S$.

The CVIs in R are added to the conic combination with the original multipliers λ_r. Those with $i_r \notin S$ and $|C_r \cap S| = 1$ are modified by adding vertex w to the clique C_r. The resulting CVI (still of the form (14) or (16)) is:

$$-x_{i_r} + \sum_{k \in C_r \cup \{w\} : \{i_r, k\} \in \bar{E}^+} x_{i_r k} \leq 0. \tag{20}$$

Again, the same construction is applied for each $z \in Z$.

The CPIs in T are included in the conic combination with (original) multipliers λ_t. These are modified if S intersects $C_t \cup C'_t$. Specifically, if S intersects only one clique, say C_t, vertex w is inserted in C_t:

$$\sum_{i \in C_t \cup \{w\} \cup C'_t} x_i - \sum_{\{i,k\} \in \bar{E}^+(C_t \cup \{w\} : C'_t)} x_{ik} \leq 1; \tag{21}$$

Else, if $|C_t \cap S| = |C'_t \cap S| = 1$ vertex w is inserted both in C_t and C'_t:

$$\sum_{i \in C_t \cup C'_t \cup \{w\}} x_i - \sum_{\{i,k\} \in \bar{E}(C_t : C'_t)} x_{ik} \leq 1. \tag{22}$$

Clearly, all these CPIs are of the form (13).

Let us now look at the variable coefficients in the conic combination. The coefficient of x_{iw}, for each $i \in V \backslash (S \cup V(S))$, is:

$$\sum_{k \in S} \left(\sum_{r \in R_{ik}} \lambda_r + \sum_{z \in Z_{ik}} \lambda_z - \sum_{t \in T_{ik}} \lambda_t \right),$$

which, by Eq. (18), is equal to zero. The coefficient of variable x_w is

$$\sum_{t \in \bigcup_{i \in S} T_i} \lambda_t - \sum_{r \in R : i_r \in S, C_r \cap S = \emptyset} \lambda_r - \sum_{z \in Z : i_z \in S, C_z \cap S = \emptyset} \lambda_z,$$

equivalent to

$$\sum_{i \in S} \left(\sum_{t \in T_i} \lambda_t - \sum_{r \in R_i} \lambda_r - \sum_{z \in Z_i} \lambda_z \right) - \sum_{\{i,k\} \subset S} \left(\sum_{r \in R_{ik}} \lambda_r + \sum_{z \in Z_{ik}} \lambda_z - \sum_{t \in T_{ik}} \lambda_t \right).$$

By Eqs. (17) and (18), this is equal to $\sum_{i \in S} a_i$, as required. The coefficients of all other variables are unchanged, and so is the right hand side b. $\square$

In summary, Theorems 1–4 show that most of the theoretical results proved for $\mathrm{N}(K, K)$ still hold for $\cap_{j \in V} \mathrm{N}_j(K, K)$. The whole picture of the strong relaxations is illustrated in Table 3.

6 Implementation

Several implementation issues have to be dealt with when generating cutting planes from $\cap_{j \in V} \mathrm{N}_j(K, K)$ by solving a CGLP. One major concern is to detect a collection Ω of tractable size, as generating (and storing) all the cliques of G is clearly impractical. On the other hand, deriving the CGLP from a subset of the cliques may return weaker cuts. In our implementation Ω is dynamically generated by a separation heuristic for clique inequalities, as explained below.

Let $\mathcal{P}$ be the current formulation and x^* the associated (optimal) fractional point. We denote by $\Omega^{\mathcal{P}}$ the collection of cliques associated to the clique inequalities in $\mathcal{P}$ and by $\Omega^{(\mathcal{P}, x^*)}$ the subset of the cliques (almost) tight to x^*, i.e., the cliques $C \in \Omega^{\mathcal{P}}$ such that $(1 - \sum_{i \in C} x_i^*) < \sigma$, $\sigma \in \mathbb{R}^+$.

Initial formulation The clique inequalities associated to any set of cliques covering all edges of G provide a valid formulation for the SSP. In our experience, computing a clique-cover of G by a greedy heuristic allows to quickly obtain a good initial formulation.

Clique-based cutting plane The upper bound UB_{clq}, introduced in Sect. 2 (see Tables 1, 2), has been computed by a cutting plane algorithm based on clique inequalities. We implemented a separation routine [14] that is an advanced, yet efficient, variant of the greedy-like heuristic introduced in [18] and also used in [31]. Clique inequalities violated by at least $\epsilon = 10^{-5}$ are then added to $\mathcal{P}$. The cutting plane algorithm is also used in the lift-and-project cut generation.

CGLP set-up For a given $j \in V$, the CGLP associated to relaxation $N_j(K, K)$ (see the development of Sect. 4) is built from the collection $\Omega^{(\mathcal{P},x^*)}$. The parameter σ is chosen in the range [0.1, 0.5] for sparse graphs (i.e., graph with density $\leq$ 15%), while for dense graphs it is enough to consider tighter cliques (i.e., $\sigma \leq 10^{-3}$).

Cut generation algorithm Cuts valid for $\cap_{j\in V} \mathrm{N}_j(K, K)$ are generated by applying independently the separation procedure to each $\mathrm{N}_j(K, K)$, $j \in V$. The algorithm performs *n_iter* iterations: in each iteration a lift-and-project cut for each $j \in V$ is (if possible) generated. The whole cut generation procedure for $\cap_{j\in V} \mathrm{N}_j(K, K)$ is described in Algorithm 1.

Algorithm 1 Cut generation algorithm

```
Input:   A formulation $\mathcal{P}$, an integer $n\_iter$;
Output:  An updated formulation $\mathcal{P}$
for $i := 1$ to $n\_iter$ do
  for $j := 1$ to $|V|$ do
    Optimize over $\mathcal{P}$, get $x^*$
    Setup and solve the CGLP for $N_j$;
    if a violated cut $a^T x \leq b$ is found then
      $\mathcal{P} \leftarrow \mathcal{P} \cup \{a^T x \leq b\}$
      Optimize over $\mathcal{P}$, get $x^*$;
      Execute clique cutting plane;
    end if
  end for
end for
```

Every time a violated cut from the CGLP has been added the clique cutting plane algorithm is executed. This is crucial to avoid that the next CGLP returns a clique inequality not in $\Omega^{\mathcal{P}}$.

The vertex sequence in the inner loop does not appear to be a crucial issue, although slightly better performance can be observed if the vertices are ranked by non-increasing degree.

All algorithms are implemented within the IBM CPLEX 11.2 framework. The CGLPs are solved by the interior point method `hybbaropt` (default settings) that significantly outperforms simplex methods. The computations were run on a machine equipped by 2 Intel Xeon 5150 processors clocked at 2.6 GHz and having 8 GB of RAM. However, all experiments have been performed in a single thread mode.

7 Computational results

The purpose of this section is to demonstrate that lift-and-project cuts generated to optimize over $\cap_{j\in V} N(K, K)$, denoted by $\mathcal{C}(N)$-cuts, do help bridge the gap between quality of the upper bound and computational tractability of the relaxation. Two experiments are presented. In the first experiment, described in Sect. 7.1, the upper bound achievable by a cutting plane algorithm based on $\mathcal{C}(N)$-cuts is compared to the other representative (either LP or SDP) bounds. In the second experiment, described in Sect. 7.2, the performance of the $\mathcal{C}(N)$-cuts in a branch-and-cut framework is investigated.

The test bed consists of three families of graphs:

– graphs from the DIMACS Second Challenge [19] available at the web site [30]. We consider the 19 out of 34 instances with $n < 400$ which are not solved to optimality by clique inequalities (see Table 1) and the graph `mann_a45`;
– uniform random graphs tested in [12] and [9], downloadable from [36] in complemented version. Graph $x.y$ has $|V| = x$ and percentage density y;
– very sparse uniform random graphs, tested in [12] and generated with the same parameters as [16].

All instances are unweighted, as they tend to be the most difficult in practice.

7.1 Comparison among upper bounds

According to the overview presented in Sects. 1 and 2, representative upper bounds for $\alpha(G)$ are: UB_{clq}, $\theta(G)$ and $UB_{\text{M}(K,K)}$, i.e., the bound obtained by optimizing over $\text{M}(K, K)$ [12]. These give reference values achievable with standard polyhedral combinatorics techniques, semidefinite relaxations and linear lifting operators respectively. In Tables 4, 6 and 8, these three upper bounds are compared with the upper bound $UB_{\mathcal{C}(N)}$, computed by Algorithm 1. Tables 5, 7 and 9 complete the picture with several computational details. For each upper bound, we report the number of inequalities of the corresponding relaxation and the CPU time spent in bound evaluation. For $\mathcal{C}(N)$ the number of "pure" $\mathcal{C}(N)$-cuts (that is, those different from clique inequalities) and the number of clique inequalities are reported separately; the number of $\mathcal{C}(N)$-cut iterations is also included. Let us discuss the experimental findings for each family of graphs.

DIMACS challenge graphs Table 4 shows that $UB_{\mathcal{C}(N)}$ is fairly close to $UB_{\text{M}(K,K)}$, which reveals that optimizing over $\cap_j \text{N}_j(K, K)$ instead of $\text{M}(K, K)$ does not significantly weaken the bound. $UB_{\mathcal{C}(N)}$ turns out to be much stronger than UB_{clq} and, most notably, outperforms $\theta(G)$ on `C.125.9`, `c-fat200-5`, `DSJC125.1`, `mann_a9`, `mann_a27`, `mann_a45`, `hamming6-4` and coincides with it on `san200_0.7-2`. It is worthwhile mentioning that no such a clear improvement on $\theta(G)$ could be accomplished even with the quite strong SDP relaxations investigated in [16,9] and [5] (see also [12] for a summary of results).

Furthermore, one can infer from Table 5 that such a strong bound is now accessible in a short time by a relatively small number of inequalities. CPU time is in most cases considerably shorter than that required to compute $UB_{\text{M}(K,K)}$.

A careful look at Table 5 also reveals a surprisingly long CPU time elapsed for graph `keller4`, especially if compared to larger graphs (even with similar densities)

Table 4 Upper bound comparison: DIMACS graphs

Graph	$\|V\|$	$\|E\|$	$\alpha(G)$	UB_{clq}	$\theta(G)$	$UB_{\mathrm{M}(K,K)}$	$UB_{\mathcal{C}(N)}$
brock200_1	200	5,066	21	38.06	27.50	30.25	33.59
brock200_2	200	10,024	12	21.33	14.22	16.09	18.27
brock200_3	200	7,852	15	27.34	18.82	21.16	23.55
brock200_4	200	6,811	17	30.67	21.29	23.80	26.77
C.125.9	125	787	34	43.06	37.89	36.53	37.81
C.250.9	250	3,141	44	71.38	56.24	59.96	63.95
c-fat200-5	200	11,427	58	66.67	60.34	58.00	58.00
DSJC125.1	125	736	34	43.15	38.39	36.99	38.22
DSJC125.5	125	3,891	10	15.44	11.47	11.41	13.21
mann_a9	45	72	16	18.50	17.47	16.85	17.11
mann_a27	378	702	126	135.00	132.76	131.39	132.44
mann_a45	1,035	1,980	345	360.00	356.04	–	355.86
hamming6-4	64	1,312	4	5.33	5.33	4.00	4.64
keller4	171	5,100	11	14.82	14.01	13.17	14.29
p_hat300-1	300	33,917	8	15.30	10.10	11.40	13.45
p_hat300-2	300	22,922	25	33.59	27.00	30.00	30.73
p_hat300-3	300	11,460	36	54.36	41.16	47.32	49.79
san200_0.7-2	200	5,970	18	19.04	18.00	18.00	18.00
sanr200_0.7	200	6,032	18	33.39	23.80	26.12	29.45
sanr200_0.9	200	2,037	42	59.82	49.30	50.73	54.52

with much larger clique collections (and, therefore, larger CGLPs). This is due to a quite slow convergence of Cplex `hybbaropt` algorithm.

Another issue to be remarked is that a larger number of clique inequalities are generated by Algorithm 1. This is a relevant side-effect of adding deep lift-and-project cuts, as these additional clique inequalities could not be discovered without "pushing forward" the current fractional point.

Uniform random graphs The results, presented in Tables 6 and 7, substantially confirm the previous observations. $UB_{\mathcal{C}(N)}$ outperforms $\theta(G)$ on `100.10` and `150.75` and coincides with it on `100.90` and `150.90`. Table 7 shows that for 10 out of 20 instances the time saving with respect to computing $UB_{\mathrm{M}(K,K)}$ exceeds one order of magnitude.

Very sparse random graphs Here graph densities do not exceed 5%. In this setting $\mathcal{C}(N)$-cuts are exceptionally effective, yielding in all cases upper bounds lower than $\theta(G)$ (Table 8). In fact, these are very close to $UB_{\mathrm{M}(K,K)}$.

Overall, the results give evidence of the relevant contribution of $\mathcal{C}(N)$-cuts in closing the integrality gap by a fairly small number of cuts. Remarkably, we observed that the very first iterations of Algorithm 1 close a relevant portion of the gap. One can

Table 5 Cutting plane details: DIMACS graphs

Graph	UB_{clq}		$UB_{\text{M}(K,K)}$		$UB_{\mathcal{C}(N)}$			
	Clique ineq.	Time	M(K, K) ineq.	Time	$\mathcal{C}(N)$ ineq.	Clique ineq.	Time	*n_iter*
brock200_1	1,997	0.34	737,410	17,670	152	3,151	373	2
brock200_2	3,924	3.02	712,705	26,501	30	5,624	190	1
brock200_3	2,861	1.41	786,606	22,386	64	4,297	338	1
brock200_4	2,542	1.04	823,317	25,362	116	4,142	196	1
C.125.9	486	0.01	286,800	227	625	517	391	5
C.250.9	1,722	0.12	1,549,908	9,397	500	1,952	8,908	2
c-fat200-5	7,561	0.25	345,779	265	68	7561	45	1
DSJC125.1	460	0.01	270,201	274	622	468	297	5
DSJC125.5	1,540	0.47	222,677	377	25	2,368	27	1
mann_a9	48	0.01	3,744	0.41	90	48	0.26	1
mann_a27	468	0.01	432,969	393	302	468	120	1
mann_a45	1,320	0.01	–	–	314	1,320	1,062	1
hamming6-4	161	0.01	9,934	4	22	412	5	2
keller4	877	0.15	696,634	15,324	171	1,897	9,586	1
p_hat300-1	7,197	30.62	526,513	4,910	24	9,800	767	1
p_hat300-2	3,280	2.28	679,808	24,337	233	5,618	2,207	1
p_hat300-3	3,968	1.68	985,233	46,408	407	5,941	2,419	2
san200_0.7-2	1,480	0.35	48,057	300	195	2,291	151	2
sanr200_0.7	2,352	0.70	1,424,109	9,971	297	4,726	762	1
sanr200_0.9	1,170	0.04	757,854	8,483	200	1,264	949	1

also observe that, unlike what typically happens with Benders cuts, $\mathcal{C}(N)$-cuts tend to be surprisingly sparse and seldom show nasty coefficients. This does not hold when cuts are generated from N(K, K) (see Sect. 3).

All these facts suggest that $\mathcal{C}(N)$-cuts may be effective in a branch-and-cut framework. This is indeed the case, as demonstrated in the next section.

7.2 Branch-and-cut results

In this section we show that, despite the computational burden of solving the CGLP, embedding Algorithm 1 into the CPLEX 11.2 branch-and-cut framework is cost-effective. Three algorithms are compared:

1. The CPLEX default algorithm;
2. A branch-and-cut algorithm (`Clique-B&C`) in which clique cuts are generated by our separation heuristic;
3. A branch-and-cut algorithm ($\mathcal{C}(N)$`-B&C`) in which lift-and-project cuts are generated by Algorithm 1.

Table 6 Upper bounds comparison: random graphs from [9]

Graph	$\|V\|$	$\|E\|$	$\alpha(G)$	UB_{clq}	$\theta(G)$	$UB_{\text{M}(K,K)}$	$UB_{\mathcal{C}(N)}$
100.10	100	490	31	37.18	33.16	31.76	32.75
100.25	100	1,216	17	23.19	19.49	19.03	20.70
100.50	100	2,419	9	13.86	10.82	10.58	11.79
100.75	100	3,710	5	7.34	5.82	5.47	6.07
100.90	100	4,463	4	4.16	4.00	4.00	4.00
150.10	150	1,096	37	48.88	41.99	41.56	43.12
150.25	150	2,724	19	31.48	24.33	25.25	27.91
150.50	150	5,510	10	18.14	12.90	13.61	15.20
150.75	150	8,373	6	9.54	6.86	6.86	6.64
150.90	150	10,038	5	5.22	5.00	5.00	5.00
200.10	200	1,958	42	61.19	50.14	51.28	53.55
200.25	200	4,851	22	39.34	28.68	31.21	35.07
200.50	200	9,874	11	21.80	14.68	16.57	18.09
200.75	200	14,801	7	11.56	7.81	8.34	9.66
200.90	200	17,853	4	5.97	4.44	4.66	5.16
250.10	250	2,998	46	73.44	58.06	62.18	67.36
250.25	250	7,584	23	45.78	31.83	37.20	40.60
250.50	250	15,457	11	25.49	16.19	19.52	23.30
250.75	250	23,199	7	14.11	8.53	9.89	11.91
250.90	250	27,976	4	7.01	4.80	5.25	5.89

In `Clique-B&C` and in $\mathcal{C}(N)$-`B&C` the CPLEX cut generation procedures and the `dynamic search` option are turned off, while the direction of the first branch is set to `up branch first`. The clique separation heuristic is invoked only if the depth of the current subproblem is less than k ($k = 5$ for graphs with density $< 30\%$, $k = 2$ otherwise) and the branching variable is fixed to 1. Exactly one iteration of Algorithm 1 in $\mathcal{C}(N)$-`B&C` is executed at the root node with $\sigma \in [10^{-3}, 10^{-5}]$. Finally, all cut generation algorithms are implemented by the `CPXcutcallbackadd` procedure, that is, the cut pool management is completely left to CPLEX. It is important to remark that `Clique-B&C` is a quite strong competitor, as it often outperforms dedicated branch-and-cut algorithms [31] and [29], that embed several separation routines, dedicated cut pool management, and specialized branching strategies.

Tables 10 and 11 report the CPU time (in seconds) and number of enumerated subproblems (on DIMACS and random graphs respectively) for the three algorithms, where a time limit of 4 hours is imposed. Random graphs with density $> 50\%$ and some very sparse random graphs are not included, as they are quickly solved by CPLEX enumeration and do not provide information.

From Tables 10 and 11 one observes that $\mathcal{C}(N)$-`B&C` always outperforms the other algorithms in terms of tree size. For DIMACS graphs (Table 10) in nine cases (out of 20) this does not lead to a time saving. However, such cases corresponds to the easiest instances, where a great effort for improving the upper bounds is not justified. On the

Table 7 Cutting plane details: random graphs from [9]

Graph	UB_{clq}		$UB_{\text{M}(K,K)}$		$UB_{\mathcal{C}(N)}$			
	Clique ineq.	Time	M(K, K) ineq.	Time	$\mathcal{C}(N)$ ineq.	Clique ineq.	Time	*n_iter*
100.10	331	0.01	135,834	92	496	339	66	5
100.25	541	0.03	688,960	364	288	679	81	5
100.50	952	0.09	620,566	3,408	92	2,095	55	1
100.75	1,327	0.35	161,008	4,381	34	3,412	29	1
100.90	1,402	0.70	184,426	296	98	1,402	24	1
150.10	676	0.01	627,802	1,977	750	708	752	5
150.25	1,171	0.11	1,047,928	1,918	426	1,801	671	5
150.50	2,207	0.72	927,532	3,523	121	5,624	366	1
150.75	3,874	4.52	1,564,190	4,921	9	8,190	127	1
150.90	4,634	5.95	130,894	5	149	4,634	739	1
200.10	1,124	0.04	1,761,137	15,777	1,000	1,232	4,425	5
200.25	2,028	0.27	1,332,884	10,948	336	3,238	573	5
200.50	3,953	2.61	787,324	4,903	36	5,766	143	1
200.75	9,350	24.30	1,265,892	3,274	8	12,930	205	1
200.90	10,341	34.31	663,423	1,126	3	14,651	50	1
250.10	1,679	0.11	1,636,414	16,225	216	1,883	432	5
250.25	3,055	0.91	1,108,108	22,201	764	5,222	3,045	5
250.50	6,497	9.39	677,402	23,011	32	7,878	240	1
250.75	16,113	40.21	809,039	623	11	19,879	171	1
250.90	25,925	82.68	909,830	188	4	34,150	122	1

Table 8 Upper bounds comparison: very sparse random graphs

Graph	$\|V\|$	$\|E\|$	$\alpha(G)$	UB_{clq}	$\theta(G)$	$UB_{\text{M}(K,K)}$	$UB_{\mathcal{C}(N)}$
150.4	150	459	58	67.50	62.40	60.21	60.80
150.5	150	556	55	62.00	58.01	55.33	56.19
170.3	170	451	70	79.50	73.51	70.00	70.00
200.2	200	420	93	97.50	94.77	93.00	93.00
200.3	200	603	80	89.00	83.63	80.38	81.13
300.2	300	905	121	142.00	128.10	123.80	124.62
350.2	350	1,206	132	156.00	141.94	137.77	139.25
400.1	400	816	187	199.00	191.42	187.00	187.19

contrary, the time saving with respect to `Clique-B&C` is significant for some of the hardest instances, namely: 13% for `phat300_3`, 23% for `sanr200_09`, 42% for `mann_a45`.

The results for random graphs (Table 11) highlights that the lift-and-project cuts are quite effective for sparse graphs. When densities go below 5%, `Clique-B&C`

Table 9 Cutting plane details: very sparse random graphs

Graph	UB_{clq}		$UB_{\text{M}(K,K)}$		$UB_{\mathcal{C}(N)}$			
	Clique ineq.	Time	M(K, K) ineq.	Time	$\mathcal{C}(N)$ ineq.	Clique ineq.	Time	n_iter
150.4	398	0	208,524	557	705	398	95	5
150.5	440	0	231,585	660	720	440	148	5
170.3	413	0	115,748	23	336	413	72	5
200.2	394	0	96,406	15	356	394	80	5
200.3	534	0	376,927	3,955	948	534	233	5
300.2	848	0	909,963	13,906	1,435	848	904	5
350.2	1,082	0	694,174	8,435	1,548	1,082	1,798	5
400.1	803	0	465,437	101	953	803	915	3

Table 10 DIMACS graphs: branch-and-cut results

Graph	CPLEX default		Clique-B&C		$\mathcal{C}(N)$-B&C	
	Time	Subprob.	Time	Subprob.	Time	Subprob.
brock200_1	1,691.88	303,352	1,153.54	119,613	**1,096.41**	**99,078**
brock200_2	118.70	9,808	91.42	2,498	**91.25**	**2,340**
brock200_3	202.18	19,461	233.10	8,239	**197.00**	**5,983**
brock200_4	484.30	71,580	435.08	22,488	**384.14**	**19,368**
C.125.9	6.27	3,458	**3.51**	3,291	7.54	**2,783**
C.250.9	+++	+++	+++	+++	+++	+++
c-fat200-5	12.11	47	**7.04**	47	48.73	**0**
DSJC125.1	7.07	4,887	**3.16**	3,805	5.76	**3,456**
DSJC125.5	11.00	1,626	**7.93**	412	9.24	**305**
mann_a9	**0.02**	3	0.02	7	0.28	**1**
mann_a27	1.87	1,888	**1.57**	6,961	2.19	**6,185**
mann_a45	109.45	67,300	61.09	55,044	**35.47**	**20,214**
hamming6-4	0.11	6	**0.09**	5	0.40	**1**
keller4	**23.40**	5,814	27.86	2,713	38.99	**2,602**
p_hat300-1	**136.23**	4549	245.10	**2,736**	259.12	2,736
p_hat300-2	210.02	5,773	94.31	1,345	**92.53**	**1,258**
p_hat300-3	+++	+++	4,040.84	79,090	**3,509.32**	**74,319**
san200_0.7-2	6.34	517	**4.07**	170	12.21	**0**
sanr200_0.7	1,008.83	161,673	689.58	34,250	**675.25**	**30,885**
sanr200_0.9	2,795.50	853,581	1,357.64	395,538	**1,039.78**	**278,983**

Bold values indicate the best performance

gets worse, as most of the cliques correspond to edges. Interestingly, in these cases, CPLEX default gomory and zero-half cuts become more effective. However, $\mathcal{C}(N)$-B&C largely outperforms the other algorithms showing up to one order magnitude subproblem and more than 50% time savings (350.2).

Table 11 Random graphs: branch-and-cut results

Graph	Cplex default		Clique cuts		$\mathcal{C}(N)$-cuts	
	Time	Subprob.	Time	Subprob.	Time	Subprob.
150.10	60.03	30,856	33.55	20,468	**33.05**	**15,753**
150.25	118.34	37,313	**80.85**	15,611	82.07	**14,452**
150.50	**25.65**	3,613	36.84	1,365	31.26	**1,327**
200.10	4,382.40	1,363,378	2,144.67	670,667	**1,772.01**	**538,902**
200.25	2,014.49	396,785	1,027.56	107,697	**1,018.57**	**79,669**
200.50	**147.59**	11,754	289.88	5,472	243.17	**4,271**
250.10	+++	+++	+++	+++	+++	+++
250.25	+++	+++	+++	+++	+++	+++
250.50	**934.49**	61,076	2,124	31,473	2,171	**30,734**
200.3	3.94	983	**1.32**	2,365	5.68	**809**
300.2	166.23	44,261	151.16	247,322	**108.79**	**29,723**
350.2	5,457.14	1,394,187	7,946.75	5,517,857	**3,346.76**	**539,245**

Bold values indicate the best performance

Tables 12 and 13 report the number of clique cuts generated by `Clique-B&C` and that of $\mathcal{C}(N)$ cuts generated by $\mathcal{C}(N)$-`B&C` (in brackets clique cuts generated at the root node). One can observe that, although a similar total number of clique inequalities are generated by the two algorithms, a larger number of them is detected by $\mathcal{C}(N)$-`B&C` at the root node. This may let some "important" clique inequalities be available at the early stage of the enumeration. The parameters σ and ϵ, which control the size of Ω have been assigned with reference values for sparse and dense graphs. However, we experienced that an instance-specific setting can indeed give some improvement.

In summary, even though the lift-and-project cuts come at the price of solving a (potentially huge) CGLP, our basic implementation turns out to be cost-effective. We believe that this experience demonstrates that these cuts represent an important ingredient to be considered when implementing a state-of-the-art branch-and-cut algorithm for the SSP.

8 Conclusions

We showed that optimizing over the new relaxation $\cap_j \mathrm{N}_j(K, K)$ yields strong and numerically robust Benders cutting planes. An open line of research deals with investigating a more advanced implementation of the cut generation algorithm, possibly exploiting some structural properties of CGLPs. This could help generate cuts more aggressively (i.e., increase $|\Omega|$, n_iter) as well as tackle larger graphs. Finally, even if some problem structure has been exploited in order to identify the new relaxation, the proposed lift-and-project method looks adaptable to other 0–1 Programming problems.

Table 12 DIMACS graphs: branch-and-cut details

Graph	Clique-B&C		$\mathcal{C}$(N)-B&C		
	# Clique inequalities	Separation time	# $\mathcal{C}$(N) ineq.	# Clique inequalities	Separation time
brock200_1	3,089 (1,997)	1.56	80	2,926 (2,129)	80.48
brock200_2	3,924 (3,924)	3.02	14	4,028 (4,028)	6.61
brock200_3	4,413 (2,861)	2.87	10	4,460 (2,911)	6.75
brock200_4	4,007 (2,542)	2.52	29	3,971 (2,646)	12.38
C.125.9	556 (486)	0.07	49	604 (506)	3.99
c-fat200-5	7,561 (7,561)	0.25	116	10,929 (10,929)	47.59
DSJC125.1	530 (460)	0.14	31	487 (466)	2.31
DSJC125.5	1,912 (1,540)	0.16	28	1,981 (1,611)	2.13
mann_a9	48 (48)	0.01	90	48 (48)	0.26
mann_a27	468 (468)	0.47	3	468 (468)	1.22
mann_a45	1,320 (1,320)	0.01	5	1,320 (1,320)	9.09
hamming6-4	161 (161)	0.01	17	310 (310)	0.29
keller4	1,340 (877)	0.32	2	1,219 (907)	15.44
p_hat300-1	7,197 (7,197)	30.67	0	7,197 (7,197)	45.13
p_hat300-2	3,597 (3,280)	4.00	3	3,592 (3,289)	11.01
p_hat300-3	5,072 (3,968)	5.86	15	4,862 (4,001)	46.89
san200_0.7-2	1,510 (1,480)	1.03	31	1,670 (1,670)	11.86
sanr200_0.7	3,290 (2,352)	1.92	51	3,331 (2,471)	25.59
sanr200_0.9	1,259 (1,170)	0.31	56	1,263 (1,217)	14.33

Table 13 Random graphs: branch-and-cut details

Graph	Clique-B&C		$\mathcal{C}$(N)-B&C		
	# Clique inequalities	Separation time	# $\mathcal{C}$(N) ineq.	# Clique inequalities	Separation time
150.10	701 (676)	0.10	32	701 (689)	4.70
150.25	1,488 (1,171)	0.34	17	1,560 (1,201)	4.15
150.50	2,893 (2,207)	0.37	20	2,973 (2,285)	2.66
200.10	1,194 (1,124)	0.20	45	1,215 (1,140)	14.41
200.25	3,056 (2,028)	1.03	22	2,741 (2,078)	10.52
200.50	5,229 (3,953)	3.63	2	5,502 (3,994)	4.97
250.50	11,471 (6,497)	13.06	60	11,587 (6,602)	38.49
200.3	534 (534)	0	50	534 (534)	4.33
300.2	848 (848)	0	103	848 (848)	32.21
350.2	1,086 (1,082)	0	314	1,475 (1,082)	416.02

References

1. Applegate, D.L., Bixby, R.E., Chvàtal, V., Cook, W.J.: The Traveling Salesman Problem: A Computational Study. Princeton University Press, Princeton (2007)
2. Balas, E., Ceria, S., Cornuéjols, G.: A lift-and-project cutting plane algorithm for mixed 0–1 programs. Math. Program. **112**, 295–324 (1993)
3. Balas, E., Ceria S., Cornuéjols, G.E., Pataki, G.: Polyhedral methods for the maximum clique problem. In: Johnson, D.S., Trick, M.A. (eds.) Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, October 11–13, 1993, Providence (1996)
4. Benders, J.F.: Partitioning procedures for solving mixed-variables programming problems. Numerische Mathematik **4**, 238–252 (1962)
5. Burer, S., Vandenbussche, D.: Solving lift-and-project relaxations of binary integer programs. SIAM J. Opt. **16**, 726–750 (2006)
6. Cheng, E., Cunningham, S.: Wheel inequalities for stable set polytopes. Math. Program. **77**, 348–421 (1997)
7. Cheng, E., De Vries, S.: Antiweb-wheel inequalities and their separation problems over the stable set polytopes. Math. Program. **92**, 153–175 (2002)
8. Cornuéjols, G.: Valid inequalities for mixed integer linear programs. Math. Program. Ser. B **112**, 3–44 (2008)
9. Dukanovic, I., Rendl, F.: Semidefinite programming relaxations for graph coloring and maximal clique problems. Math. Program. **109**, 345–365 (2007)
10. Fischetti, M., Salvagnin, D., Zanette, A.: On the Selection of Benders' cuts. Math. Program. Ser. B **124**, 175–182 (2008)
11. Giandomenico, M., Letchford, A.N.: Exploring the relationship between max-cut and stable set relaxations. Math. Program. **106**, 159–175 (2006)
12. Giandomenico, M., Letchford, A.N., Rossi, F., Smriglio, S.: An application of the Lovász-Schrijver $M(K, K)$ operator to the stable set problem. Math. Program. Ser. A **120/2**, 381–401 (2009)
13. Giandomenico, M., Letchford, A.N., Rossi, F., Smriglio, S.: A new approach to the stable set problem based on ellipsoids. Accepted for presentation at IPCO XV (2011)
14. Giandomenico, M., Rossi, F., Smriglio, S.: Implementing separation heuristics for clique inequalities, working paper
15. Grötschel, M., Lovász, L., Schrijver, A.J.: Geometric Algorithms and Combinatorial Optimization. Springer, New York (1988)
16. Gruber, G., Rendl, F.: Computational experience with stable set relaxations. SIAM J. Opt. **13**, 1014–1028 (2003)
17. Håstad, J.: Clique is hard to approximate within $n^{1-\epsilon}$. Acta Math. **182**, 105–142 (1999)
18. Hoffman, K.L., Padberg, M.: Solving airline crew scheduling problems by branch-and-cut. Manage. Sci. **39**(6), 657–682 (1993)
19. Johnson, D.S., Trick, M.A. (eds.): Cliques, Coloring and Satisfiability: The 2nd DIMACS Implementation Challenge. American Mathematical Society, Providence (1996)
20. Juhász, F.: The asymptotic behaviour of Lovász' θ function for random graphs. Combinatorica **2**, 153–155 (1982)
21. Lovász, L.: On the Shannon capacity of a graph. IEEE Trans. Inf. Theor. **25**, 1–7 (1979)
22. Lovász, L., Schrijver, A.J.: Cones of matrices and set-functions and 0–1 optimization. SIAM J. Opt. **1**, 166–190 (1991)
23. Malik, J., Pohv, J., Rendl, F., Wiegele, A.: Regularization methods for semidefinite programming. SIAM J. Opt. **20/**1 (2009)
24. Mannino, C., Sassano, A.: An exact algorithm for the maximum stable set problem. Comput. Optim. Appl. **3**, 243–258 (1994)
25. Mannino, C., Sassano, A.: Edge projection and the maximum cardinality stable set problem. In: Johnson, D.S., Trick, M.A. (eds.) Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, October 11—13, 1993, Providence (1996)
26. Nemhauser, G.L., Sigismondi, G.: A strong cutting plane/branch-and-bound algorithm for node packing. J. Oper. Res. Soc. **43**, 443–457 (1992)
27. Padberg, M.W.: On the facial structure of set packing polyhedra. Math. Program. **5**, 199–215 (1973)
28. Pohv, J., Rendl, F., Wiegele, A.: A boundary point method to solve semidefinite programs. Computing **78**, 277–286 (2006)

29. Rebennack, S., Oswald, M., Theis, D.O., Seitz, H., Reinelt, G., Pardalos, P.M.: A Branch and cut solver for the maximum stable set problem. J. Comb. Optim. doi:10.1007/s10878-008-9175-8
30. Repository. ftp://dimacs.rutgers.edu/pub/challenge/graph/benchmarks/clique
31. Rossi, F., Smriglio, S.: A branch-and-cut algorithm for the maximum cardinality stable set problem. Oper. Res. Lett. **28**(2), 63–74 (2001)
32. Sewell, E.C.: A branch and bound algorithm for the stability number of a sparse graph. INFORMS J. Comput. **10**(4), 438–447 (1998)
33. Sherali, H.D., Adams, W.P.: A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. SIAM J. Discret. Math. **3**, 411–430 (1990)
34. Tomita, E., Kameda, T.: An efficient branch-and-bound algorithm for finding a maximum clique, with computational experiments. J. Glob. Opt. **37**, 95–111 (2007)
35. Trotter, L.E.: A class of facet-producing graphs for vertex packing polyhedra. Discret. Math. **12**, 373–388 (1975)
36. Website. http://www.math.uni-klu.ac.at/or/Software/software.html
37. Wilson, A.T.: Applying the boundary point method to an SDP relaxation of the maximum independent set problem for a branch and bound algorithm, Master thesis, New Mexico Institute of mining and technology (2009)