Ellipsoidal Relaxations of the Stable Set Problem: Theory and Algorithms

Monia Giandomenico* Adam N. Letchford†
Fabrizio Rossi* Stefano Smriglio*

April 2014

Abstract

A new exact approach to the stable set problem is presented, which attempts to avoids the pitfalls of existing approaches based on linear and semidefinite programming. The method begins by constructing an ellipsoid that contains the stable set polytope and has the property that the upper bound obtained by optimising over it is equal to the Lovász theta number. This ellipsoid can then be used to construct useful convex relaxations of the stable set problem, which can be embedded within a branch-and-bound framework. Extensive computational results are given, which indicate the potential of the approach.

Keywords: combinatorial optimisation, stable set problem, semidefinite programming.

1 Introduction

Given an undirected graph $G = (V, E)$, a *stable set* is a set of pairwise non-adjacent vertices. The *stable set problem* (SSP) calls for a stable set of maximum cardinality. The cardinality of this stable set is called the *stability number* of G and is denoted by $\alpha(G)$.

The SSP is $\mathcal{NP}$-hard [22], hard to approximate [21], and hard to solve in practice (e.g., [10, 14, 30, 34, 35, 36]). Moreover, it is a remarkable fact that sophisticated mathematical programming algorithms for the SSP, such as those in [14, 30, 34, 36], have not performed significantly better than relatively simple algorithms based on implicit enumeration, such as those in [10, 35].

*Dipartimento di Informática, Università Di L'Aquila, Via Vetoio, 67010, Coppito (AQ), Italy. E-mail: fgiandomenico, rossi, smrigliog@di.univaq.it

†Department of Management Science, Lancaster University, Lancaster LA1 4YX, United Kingdom. E-mail: A.N.Letchford@lancaster.ac.uk

A possible explanation for the failure of mathematical programming approaches is the following. Linear Programming (LP) relaxations can be solved reasonably quickly, but tend to yield weak upper bounds. Semidefinite Programming (SDP) relaxations, on the other hand, typically yield much stronger bounds, but take longer to solve. Therefore, branch-and-bound algorithms based on either LP or SDP relaxations are slow, due to the large number of nodes in the search tree, or the long time taken to process each node, respectively.

In this paper we present a potential way out of this impasse. The key concept is that one can efficiently construct an *ellipsoid* that contains the stable set polytope, in such a way that the upper bound obtained by optimising over the ellipsoid is equal to the standard SDP bound, the so-called *Lovász theta number*. This ellipsoid can then be used to construct useful convex programming relaxations of the stable set problem or, perhaps more interestingly, to derive cutting planes. These cutting planes turn out to be strong and easy to generate.

We remark that our approach can be applied to the variant of the SSP in which vertices are weighted, and one seeks a stable set of maximum weight.

The paper is structured as follows. Some relevant literature is reviewed in Section 2. The ellipsoids are analysed theoretically in Section 3. Some computational experiments are discussed in Section 4, and concluding remarks are made in Section 5.

Remark: An extended abstract of this paper appeared in the 2011 IPCO proceedings [16]. In this full version, we explore three different algorithmic schemes (see Section 4), rather than only one as in [16].

2 Literature Review

We now review the relevant literature. From this point on, $n = |V|$ and $m = |E|$.

2.1 Linear programming relaxations

The SSP has the following natural formulation as a 0-1 LP:

$$\begin{aligned} \max \quad & \textstyle\sum_{i \in V} x_i & & \\ \text{s.t.} \quad & x_i + x_j \leq 1 \quad (\{i,j\} \in E) & & (1) \\ & x \in \{0,1\}^n, & & (2) \end{aligned}$$

where the variable x_i takes the value 1 if and only if vertex i is in the stable set.

The convex hull in $\mathbb{R}^n$ of feasible solutions to (1)–(2) is called the *stable set polytope* and denoted by STAB(G). This polytope has been studied

in great depth (see, e.g., [4, 18, 32]). The most well-known facet-defining inequalities for STAB(G) are the *clique* inequalities of Padberg [32]. A *clique* in G is a set of pairwise adjacent vertices, and the associated inequalities take the form:

$$\sum_{i \in C} x_i \leq 1 \qquad (\forall C \in \mathcal{C}), \tag{3}$$

where $\mathcal{C}$ denotes the set of maximal cliques in G. Note that the clique inequalities dominate the edge inequalities (1).

The polytope:

$$\left\{x \in \mathbb{R}^n_+ : (3) \text{ hold}\right\}$$

is denoted by QSTAB(G). The upper bound on $\alpha(G)$ obtained by optimising over QSTAB(G) is called the *fractional clique covering number* and denoted by $\bar{\chi}^f(G)$. By definition, we have STAB(G) $\subseteq$ QSTAB(G) and $\alpha(G) \leq \bar{\chi}^f(G)$. Unfortunately, computing $\bar{\chi}^f(G)$ is $\mathcal{NP}$-hard, since the separation problem for clique inequalities is $\mathcal{NP}$-hard [31]. On the other hand, some fast and effective separation heuristics exist, not only for clique inequalities, but also for various other inequalities (e.g., [4, 14, 30, 34, 36]). Nevertheless, LP-based cutting-plane approaches can run into difficulties when n exceeds 250 or so, mainly due to the weakness of the upper bounds.

2.2 Semidefinite programming relaxations

Lovász [26] introduced another upper bound on $\alpha(G)$, called the *theta number* and denoted by $\theta(G)$, which is based on an SDP relaxation. The bound can be derived in several different ways, and we follow the derivation presented in [18]. We start by formulating the SSP as the following non-convex quadratically-constrained program:

$$\begin{aligned}
\max \quad & \textstyle\sum_{i \in V} x_i && (4)\\
\text{s.t.} \quad & x_i^2 - x_i = 0 \quad (i \in V) && (5)\\
& x_i x_j = 0 \quad (\{i,j\} \in E). && (6)
\end{aligned}$$

In order to linearise the constraints, we introduce an auxiliary matrix variable $X = xx^T$, along with the augmented matrix

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

We then note that Y is real, symmetric and positive semidefinite (psd), which we write as $Y \succeq 0$. This leads to the following SDP relaxation:

$$\begin{aligned}
\max \quad & \textstyle\sum_{i \in V} x_i && (7)\\
\text{s.t.} \quad & x = \operatorname{diag}(X) && (8)\\
& X_{ij} = 0 \quad (\{i,j\} \in E) && (9)\\
& Y \succeq 0. && (10)
\end{aligned}$$

The corresponding upper bound is $\theta(G)$. One can compute $\theta(G)$ to arbitrary fixed precision in polynomial time [18].

Lovász [26] proved that $\alpha(G) \leq \theta(G) \leq \bar{\chi}^f(G)$. (This is a remarkable result, given that it is $\mathcal{NP}$-hard to compute $\bar{\chi}^f(G)$.) In practice, $\theta(G)$ is often a reasonably strong upper bound on $\alpha(G)$ (e.g., [5, 9, 19, 34]). Unfortunately, solving large-scale SDPs can be rather time-consuming, which makes SDP relaxations somewhat unattractive for use within a branch-and-bound framework.

The above SDP can be strengthened by adding various valid inequalities (e.g., [4, 9, 13, 19, 27, 37]). We omit details, for the sake of brevity.

2.3 The Lovász theta body and ellipsoids

The following beautiful result can be found in Grötschel, Lovász & Schrijver [18]. Let us define the following convex set:

$$\mathrm{TH}(G) = \left\{ x \in \mathbb{R}^n : \exists X \in \mathbb{R}^{n \times n} : \ (8) - (10) \text{ hold} \right\}.$$

Then we have:

$$\mathrm{STAB}(G) \subseteq \mathrm{TH}(G) \subseteq \mathrm{QSTAB}(G). \tag{11}$$

This provides an alternative proof that $\alpha(G) \leq \theta(G) \leq \bar{\chi}^f(G)$.

The set $\mathrm{TH}(G)$ is called the *theta body*. A characterisation of $\mathrm{TH}(G)$ in terms of convex quadratic inequalities was given by Fujie & Tamura [12]. For a given vector $\mu \in \mathbb{R}^m$, let $M(\mu)$ denote the symmetric matrix with $\mu_{ij}/2$ in the ith row and jth column whenever $\{i, j\} \in E$, and zeroes elsewhere. Then, given vectors $\lambda \in \mathbb{R}^n$ and $\mu \in \mathbb{R}^m$ such that $\mathrm{Diag}(\lambda) + M(\mu)$ is psd, the set

$$E(\lambda, \mu) = \left\{ x \in \mathbb{R}^n : \ x^T(\mathrm{Diag}(\lambda) + M(\mu))x \leq \lambda^T x \right\} \tag{12}$$

is an ellipsoid that contains $\mathrm{STAB}(G)$. The result we need is the following:

Theorem 1 (Fujie & Tamura, 2002) *For any graph G, we have:*

$$TH(G) = \bigcap_{\lambda, \mu : \mathrm{Diag}(\lambda) + M(\mu) \succeq 0} E(\lambda, \mu).$$

2.4 Relaxations of non-convex quadratic problems

For what follows, we will also need a known result concerning relaxations of non-convex quadratic problems, mentioned for example in [11, 24, 33]. Given a problem of the form:

$$\begin{array}{ll} \inf & x^T Q^0 x + c^0 \cdot x \\ \text{s.t.} & x^T Q^j x + c^j \cdot x = b_j \quad (j = 1, \ldots, r) \\ & x \in \mathbb{R}^n, \end{array} \tag{13}$$

we can construct the following SDP relaxation:

$$\begin{array}{lll} \inf & Q^0 \bullet X + c^0 \cdot x & \\ \text{s.t.} & Q^j \bullet X + c^j \cdot x = b_j & (j = 1, \ldots, r) \\ & Y \succeq 0, & \end{array} \tag{14}$$

where $Q^j \bullet X$ denotes $\sum_{i=1}^{n} \sum_{k=1}^{n} Q^j_{ik} X_{ik}$. Alternatively, we can form a Lagrangian relaxation by relaxing the constraints (13), using a vector $\phi \in \mathbb{R}^r$ of Lagrangian multipliers. The Lagrangian objective function is then:

$$f(x, \phi) \;=\; x^T \left(Q^0 + \sum_{j=1}^{r} \phi_j Q^j \right) x + \left(c^0 + \sum_{j=1}^{r} \phi_j c^j \right) \cdot x - \sum_{j=1}^{r} \phi_j b_j,$$

and the Lagrangian dual is:

$$\inf_{\phi} \sup_{x} f(x, \phi).$$

The result that we need is as follows:

Theorem 2 (Various authors) *A vector ϕ^* is an optimal solution to the Lagrangian dual if and only if it is an optimal dual vector for the constraints (14) in the SDP. If such a vector ϕ^* exists, then the function $f(x, \phi^*)$ is convex, the matrix $Q^0 + \sum_{j=1}^{r} \phi^*_j Q^j$ is psd, and the Lagrangian dual and the SDP yield the same lower bound.*

In our recent paper [17], this result is applied to the SSP, yielding a subgradient algorithm that approximates $\theta(G)$ from above. Luz & Schrijver [28] showed that the optimal value of the Lagrangian dual remains equal to $\theta(G)$ even if one forces all of the Lagrangian multipliers for the constraints (5) to equal 1.

3 On Ellipsoids

From now on, when we say 'ellipsoid', we mean one of the ellipsoids in the family defined by Fujie & Tamura [12] (see Subsection 2.3). Note that the convex quadratic inequality in the definition (12) is a linear combination of the constraints (5), (6). This implies that every extreme point of STAB(G) lies on the boundary of every ellipsoid. In the following three subsections, we study the ellipsoids in more detail.

3.1 Optimal ellipsoids

Recall that Theorem 1 expresses TH(G) as the intersection of the entire infinite family of ellipsoids. The following proposition states that there exists at least one ellipsoid with a very desirable property.

Theorem 3 *There exists at least one optimal pair (λ^*, μ^*) of dual vectors for the constraints (8) and (9) in the SDP (7)-(10). Moreover, for any such pair, we have:*

$$\theta(G) = \max\left\{\sum_{i\in V} x_i : x \in E(-\lambda^*, -\mu^*)\right\}.$$

Proof. The SDP (7)–(10) has the form specified in Theorem 3.1 of Tunçel [38], and therefore satisfies the Slater condition. As a result, an optimal dual pair (λ^*, μ^*) exists, strong duality holds, and the cost of the dual solution is $\theta(G)$. Applying Theorem 2 in Subsection 2.4, but switching signs to take into account the fact that the objective (7) is of maximisation type, we have:

$$\theta(G) = \max\left\{\sum_{i\in V} x_i - x^T(\mathrm{Diag}(\lambda^*) + M(\mu^*))x + \lambda^* \cdot x :\ x \in \mathbb{R}^n\right\}.$$

Moreover, the matrix $-\mathrm{Diag}(\lambda^*) - M(\mu^*)$ must be psd.

Now, given that this is a concave maximisation problem, and the fact that the pair (λ^*, μ^*) are also optimal Lagrangian multipliers for the constraints (5), (6), respectively, we have:

$$\theta(G) = \max\left\{\sum_{i\in V} x_i :\ -x^T(\mathrm{Diag}(\lambda^*) + M(\mu^*))x \le -\lambda^* \cdot x,\ x \in \mathbb{R}^n\right\},$$

which proves the result. □

In other words, an optimal dual solution to the SDP can be used to construct a relaxation of the SSP that has a linear objective function and a single convex quadratic constraint, whose corresponding upper bound is equal to $\theta(G)$. Here are two simple examples:

Example 1: Let G be a stable set on n nodes, for which $\alpha(G) = \theta(G) = n$. The unique optimal dual solution has $\lambda_i^* = -1$ for all $i \in V$. The corresponding convex quadratic constraint is:

$$\sum_{i\in V} x_i^2 \le \sum_{i\in V} x_i.$$

This can be written in the alternative form

$$\sum_{i\in V} (x_i - 1/2)^2 \le n/4.$$

One then sees that it defines a sphere of radius $\sqrt{n}/2$ centred at $(1/2, \ldots, 1/2)^T$. The maximum value that $\sum_{i\in V} x_i$ can take over this sphere is n, attained

when $x_i = 1$ for all $i \in V$. $\square$

Example 2: Let G be a clique on n nodes, for which $\alpha(G) = \theta(G) = 1$. The unique optimal dual solution has $\lambda_i^* = -1$ for all $i \in V$ and $\mu_e^* = -2$ for all $e \in E$. The corresponding convex quadratic constraint is:

$$\left(\sum_{i \in V} x_i\right)^2 \leq \sum_{i \in V} x_i.$$

This is equivalent to the linear inequalities $0 \leq \sum_{i \in V} x_i \leq 1$. $\square$

Note that the "ellipsoid" in Example 2 is unbounded. We say more about this in the next subsection. We close this subsection with a less trivial example:

Example 3: Let G be the 5-hole, i.e., let $V = \{1, 2, 3, 4, 5\}$ and $E = \{\{1, 2\}, \{2, 3\}, \{3, 4\}, \{4, 5\}, \{1, 5\}\}$. In this case, $\alpha(G) = 2$ [32] and $\theta(G) = \sqrt{5}$ [18]. The unique optimal dual solution has $\lambda_i^* = -1$ for all $i \in V$ and $\mu_e^* = 1 - \sqrt{5}$ for all $e \in E$. The corresponding convex quadratic constraint is:

$$\sum_{i \in V} x_i^2 + (\sqrt{5} - 1) \sum_{\{i,j\} \in E} x_i x_j \leq \sum_{i \in V} x_i.$$

The maximum value that $\sum_{i \in V} x_i$ can take over the corresponding ellipsoid is $\sqrt{5}$, attained when $x_i = 1/\sqrt{5}$ for all $i \in V$. $\square$

3.2 Degenerate and weak ellipsoids

At this point, it is helpful to define two special kinds of ellipsoids, as follows:

Definition 1 *An ellipsoid $E(\lambda, \mu)$ will be called 'degenerate' if it has infinite volume, or, equivalently, if the matrix* $\mathrm{Diag}(\lambda) + M(\mu)$ *is psd but not positive definite.*

Definition 2 *An ellipsoid $E(\lambda, \mu)$ will be called 'weak' if there exists an integer $k > 1$ and distinct vector pairs $(\lambda^1, \mu^1), \ldots, (\lambda^k, \mu^k)$ such that*

$$\cap_{i=1}^k E(\lambda^i, \mu^i) \subseteq E(\lambda, \mu).$$

Otherwise, it will be called 'strong'.

These concepts are now illustrated on the same three examples as in the previous subsection:

Example 1 (cont.): We saw that, when G is a stable set on n nodes, the unique optimal ellipsoid is a sphere of radius $\sqrt{n}/2$ centered at $(1/2, \ldots, 1/2)^T$. This sphere is a non-degenerate ellipsoid. On the other hand, observe that,

for $i = 1, \ldots, n$, the following convex set is an ellipsoid that is both degenerate and strong:

$$\left\{x \in \mathbb{R}^n : \, x_i^2 \leq x_i\right\} = \left\{x \in \mathbb{R}^n : \, x_i \in [0,1]\right\}. \tag{15}$$

The intersection of these n degenerate and strong ellipsoids is the unit hypercube, which is strictly contained in the sphere mentioned. (In fact it is equal to STAB(G).) Therefore the optimal ellipsoid is weak in this case. □

Example 2 (cont.): We saw that, when G is a clique on n nodes, the unique optimal ellipsoid is defined by the two linear inequalities $0 \leq \sum_{i \in V} x_i \leq 1$. So, the ellipsoid is degenerate. On the other hand, the inequality $\sum_{i \in V} x_i \leq 1$ is a clique inequality, which defines a facet of STAB(G). So, the ellipsoid is strong. □

Example 3 (cont.): We saw that, when G is the 5-hole, the unique optimal dual solution has $\lambda_i^* = -1$ for all $i \in V$ and $\mu_e^* = (1 - \sqrt{5})/2$ for all $e \in E$. The minimum eigenvalue of the associated matrix $-\text{Diag}(\lambda) - M(\mu)$ is zero, which means that the optimal ellipsoid is degenerate. We do not know whether it is strong or weak, but conjecture that it is strong. □

The following additional example shows that the unique optimal ellipsoid can be both degenerate *and* weak.

Example 4: Let G be the union of an isolated node and an edge. That is, let $n = 3$ and let E contain the edge $\{2, 3\}$. For this graph, $\alpha(G) = \theta(G) = 2$. The unique optimal dual solution has $\lambda_i^* = -1$ for all $i \in V$ and $\mu_{23}^* = -2$. The corresponding convex quadratic constraint is:

$$x_1^2 + (x_2 + x_3)^2 \leq x_1 + x_2 + x_3.$$

Now observe that any point $x^* \in \mathbb{R}^3$ with $x_1^* \in \{0, 1\}$ and $x_3^* = -x_2$ lies on the boundary of this ellipsoid. Therefore the ellipsoid is degenerate. On the other hand, using the same arguments as in Example 1 and Example 2, the following four ellipsoids are both degenerate and strong:

$$\begin{aligned} &\left\{x \in \mathbb{R}^3 : \, x_i \in [0,1]\right\} \qquad (i = 1, 2, 3) \\ &\left\{x \in \mathbb{R}^3 : \, 0 \leq x_2 + x_3 \leq 1\right\}. \end{aligned}$$

The intersection of these four ellipsoids is equal to STAB(G) in this case. Therefore the optimal ellipsoid is weak. □

We will see in Section 4 that degenerate ellipsoids can cause numerical problems for our algorithms. Fortunately, we found that, for graphs of practical interest, the optimal ellipsoid is very rarely degenerate. As for weak ellipsoids, they do not cause numerical problems. This is fortunate, since we are not aware of any polynomial-time algorithm to test if an ellipsoid is weak.

3.3 Ellipsoids and perfect graphs

Given a graph G, the minimum number of cliques needed to cover the nodes of G is called the *clique covering number* and denoted by $\bar{\chi}(G)$. By definition, $\bar{\chi}^f(G) \leq \bar{\chi}(G)$, and we already saw in Subsection 2.2 that $\alpha(G) \leq \theta(G) \leq \bar{\chi}^f(G)$. A graph G is called *perfect* if $\alpha(G') = \bar{\chi}(G')$ holds for every node-induced subgraph $G' \subseteq G$.

We will prove three results concerned with ellipsoids and perfect graphs. The first is that only degenerate ellipsoids are needed to describe the stable set polytope in the perfect graph case:

Proposition 1 *If $G = (V, E)$ is a perfect graph, then STAB(G) is the intersection of a finite number of ellipsoids that are strong, but degenerate.*

Proof. When G is perfect, STAB(G) = QSTAB(G), and is therefore completely defined by clique and non-negativity inequalities. Using the same argument as in Example 2 in the previous subsections, given any clique C, the convex quadratic inequality

$$\left(\sum_{i \in C} x_i\right)^2 \leq \sum_{i \in C} x_i$$

defines the following strong and degenerate ellipsoid:

$$\left\{x \in \mathbb{R}^n : \ 0 \leq \sum_{i \in C} x_i \leq 1\right\}.$$

Similarly, using the same argument as in Example 1 in the last subsection, for each $i \in V$, the convex quadratic inequality $x_i^2 \leq x_i$ defines the following strong and degenerate ellipsoid:

$$\{x \in \mathbb{R}^n : \ x_i \in [0, 1]\}.$$

The intersection of these ellipsoids satisfies all clique and non-negativity inequalities. □

The second result is that, if one is given a minimum cardinality clique cover of a perfect graph, then one can easily construct optimal ellipsoids:

Proposition 2 *Let $G = (V, E)$ be a perfect graph, and let $C_1, \ldots, C_{\alpha(G)} \subset V$ be cliques that form a clique cover. If necessary, delete vertices from the cliques in the family until each vertex is included in exactly one clique in the family. Let $C'_1, \ldots, C'_{\alpha(G)}$ be the resulting cliques. Then the convex quadratic inequality*

$$\sum_{k=1}^{\alpha(G)} \left(\sum_{i \in C'_k} x_i\right)^2 \leq \sum_{k=1}^{\alpha(G)} \sum_{i \in C'_k} x_i \tag{16}$$

defines an optimal ellipsoid.

Proof. One can re-write the inequality (16) in the alternative form:

$$\sum_{k=1}^{\alpha(G)} \left(\sum_{i \in C'_k} x_i - \frac{1}{2} \right)^2 \leq \frac{\alpha(G)}{4}.$$

This in turn implies that

$$\sum_{k=1}^{\alpha(G)} \sum_{i \in C'_k} x_i \leq \alpha(G), \tag{17}$$

with equality if and only if $\sum_{i \in C'_k} x_i = 1$ for $k = 1, \ldots, \alpha(G)$. But the left-hand side of (17) is equal to $\sum_{i \in V} x_i$ by construction. □

Unfortunately, the ellipsoids described in Proposition 2 are almost always both degenerate and weak:

Proposition 3 *The optimal ellipsoids obtained via Proposition 2 are:*

- *degenerate unless G is a stable set;*
- *weak unless G is a clique.*

Proof. By construction, the rank of the matrix $-\text{Diag}(\lambda) - M(\mu)$, i.e., the matrix of quadratic terms in (16) is $\alpha(G)$. Therefore, the matrix is positive definite only if $\alpha(G) = n$ i.e., if G is a stable set.

For the second point, recall that Proposition 1 gave a complete characterisation of the convex quadratic inequalities needed to define strong ellipsoids for perfect graphs. The inequality (16) is one of them if and only if $\alpha(G) = 1$, i.e., if G is a clique. □

4 Using Ellipsoids Computationally

Our goal is to design branch-and-cut algorithms that exploit the existence of (near) optimal ellipsoids. In this section, we discuss ways in which this can be done.

We consider an initial linear programming formulation based on clique inequalities. In particular, it is well-known (see e.g. [14]) that the inequalities associated with a greedily computed collection $\mathcal{C}$ of cliques covering all the edges of G yield a formulation of the stable set problem with a nice compromise between quality of the linear relaxation and formulation size. Now, the constraint associated to a (near-)optimal ellipsoid (obtained by computing approximate optimal dual multipliers of problem (7)–(10)) is added to

the problem, so as to obtain a quadratically constrained program (QCP) of the form:

$$\begin{array}{ll} \max & \sum_{i \in V} x_i \\ \text{s.t.} & x^T A x \leq b^T x \\ & \sum_{i \in C} x_i \leq 1 \quad (C \in \mathcal{C}) \\ & x \in [0,1]^n. \end{array}$$

Different algorithms can be developed exploiting different representations of the quadratic constraint $x^T A x \leq b^T x$. Three such options are described in the following section.

4.1 Alternative representations of the quadratic constraint

The first option consists of converting the quadratic constraint into a *second-order conic* constraint. This can be profitable since such constraints can be handled quite efficiently via interior-point algorithms (see, e.g., [1, 25]).

If we let A be factorised as $B^T B$, this representation amounts to adding a new vector of variables $y \in \mathbb{R}^n$ and two more variables z_1 and z_2, and replacing the original constraint with:

$$\begin{aligned} Bx &= y \\ z_1 &= 1 - b^T x \\ z_2 &= 1 + b^T x \\ z_2 &\geq \left\| \binom{z_1}{y} \right\|. \end{aligned}$$

Notice that all these constraints are linear, apart from the last, which is a second-order conic constraint.

The second option is to approximate the second-order conic constraint with a set of linear constraints. In a famous paper, Ben-Tal and Nemirovski [6] showed that, given any SOCP and any positive integer d, one can replace each SOCP constraint with $\mathcal{O}(d)$ linear constraints, involving $\mathcal{O}(d)$ additional variables, in such a way that the optimal solution to the LP matches the solution to the SOCP in the first d decimal places. The reason for doing this is that one then need only solve LPs within a branch-and-bound algorithm.

The third option is to work in the space of the original x variables, and approximate the original quadratic inequality directly by means of linear inequalities. The standard way to do this is the cutting-plane method of Kelley [23]. Given a convex feasible region defined by the non-linear inequality $f(x) \leq 0$, it constructs a polyhedral outer approximation via a set of linear constraints of the form $\nabla f(\bar{x}^i) \cdot (x - \bar{x}^i) \leq 0$, for a collection of suitably chosen points $\bar{x}^i$. In our implementation, these points are chosen

to lie on the boundary of the ellipsoid, so that the linear constraints define tangent hyperplanes on the boundary of the ellipsoid.

Here are the details. Since the ellipsoid is invariably non-degenerate in practice, it has a centre, which we denote by $\hat{x}$. One can easily shown that $\hat{x} = \frac{1}{2}A^{-1}b$. Then, to generate a collection of Kelley inequalities, one simply runs Kelley's cutting-plane algorithm, using the following separation routine to generate cuts in each iteration:

1. Let $x^* \in [0,1]^n$ be a point lying outside the ellipsoid.
2. Find the value $0 < \lambda < 1$ such that the point
$$\tilde{x} = \lambda x^* + (1-\lambda)\hat{x}$$
lies on the boundary of the ellipsoid. (This can be done by solving a quadratic equation in the single real variable λ.)
3. Generate the Kelley cutting plane corresponding to $\tilde{x}$.

This approach has the advantage that the branch-and-cut algorithm needs to solve only LPs that involve the original x variables. Moreover, the outer approximation can be strengthened by adding well-known valid inequalities for STAB(G).

In particular, one can imagine improving the outer approximation by adding violated clique inequalities of the form $\sum_{i \in C} x_i \leq 1$ (for some maximal clique C in G) along with each Kelley cut generated. Note that by adding clique inequalities alone, one cannot expect a bound improvement, since TH(G) already satisfies all clique inequalities. Nevertheless, we have found that the addition of clique inequalities can improve the convergence of the cutting-plane algorithm, probably because they can be generated more quickly than Kelley cuts and are more well-behaved numerically. Algorithm 1 describes the Kelley cutting-plane algorithm with additional generation of clique inequalities.

Remark: In the extended abstract of this paper [16], we pointed out that the Kelley cuts can be strengthened using a simple "sequential lifting" procedure. In our experiments, however, we found this to be of little benefit. For the sake of brevity, we do not go into further details.

4.2 Computational experience

We now illustrate an experimental comparison of the different algorithmic options. Two major issues are addressed: (*i*) show that the θ-bound is computationally accessible by linear descriptions and (*ii*) check whether such descriptions can be cost-effective when exploited in branch-and-cut frameworks. Additional experimental indications are also discussed in order to clarify the benefits and burdens of the method.

Algorithm 1 Kelley's cutting plane algorithm with clique inequalities

Input: A linear formulation $\mathcal{P}$, an ellipsoid $E(\bar{\lambda}, \bar{\mu})$
Output: An updated formulation $\bar{\mathcal{P}}$

$\bar{\mathcal{P}} \leftarrow \mathcal{P}$
for $i := 1$ to `maxiter` **do**
  optimize over $\bar{\mathcal{P}}$, **get** x^*
  evaluate $\tilde{x}$, $\nabla f(\tilde{x})$
  if $\nabla f(\tilde{x}) \cdot (x^* - \tilde{x}) > 0$ **then**
    scale $\nabla f(\tilde{x}) \cdot (x - \tilde{x}) \leq 0$ by a constant factor $\Gamma \rightarrow a^T x \leq b$
    round down $\lfloor a^T \rfloor x \leq \lfloor b \rfloor$
    $\bar{\mathcal{P}} \leftarrow \bar{\mathcal{P}} \cup \{\lfloor a^T \rfloor x \leq \lfloor b \rfloor\}$
    optimize over $\bar{\mathcal{P}}$, **get** x^*
    execute clique cutting plane that
    returns a collection of clique inequalities $\mathcal{C}$;
    $\bar{\mathcal{P}} \leftarrow \bar{\mathcal{P}} \cup \mathcal{C}$
  end if
end for
return $\bar{\mathcal{P}}$

4.3 Test bed and implementation details

The test bed consists of the graphs from the DIMACS Second Challenge [20] with up to 700 vertices, available at the web site [8]. The instance features are reported in Table 4.3. We do not report on the "easy" graphs, that is, those quickly solved to optimality by general MIP solvers.

The cutting plane/branch-and-cut algorithms are implemented by the Gurobi Optimizer 5.6.2 framework in a `linux x86_64` architecture (Ubuntu 12.04), compiler `g++ 4.8`. Gurobi is also used as QCP solver. The computations are run on a machine with two Intel Xeon 5150 processors (for a total of 4 cores) clocked at 2.6 GHz and 8GB RAM.

In the preliminary version presented at the IPCO 2011 conference [16], the computation of the multipliers yielding the optimal ellipsoid with some desired precision was carried out by a state-of-the-art SDP solver [29]. Afterwards, we showed that a tailored implementation of the subgradient method may be much faster, if a slight impairment of the upper bound is accepted [17]. This is indeed crucial to the application of the ellipsoidal method to large graphs. Therefore, the following results are obtained by using our subgradient implementation to carry out the computation of the ellipsoid $E(-\bar{\lambda}, -\bar{\mu})$.

4.4 Strength of the linear descriptions

Table 2 compares $\theta(G)$ with four upper bounds:

Name	$\|V\|$	Density	$\alpha(G)$	$\theta(G)$
brock200_1	200	0.25	21	27.46
brock200_2	200	0.5	12	14.23
brock200_3	200	0.39	15	18.82
brock200_4	200	0.34	17	21.29
brock400_1	400	0.25	27	39.7
brock400_2	400	0.25	29	39.56
brock400_3	400	0.25	31	39.48
brock400_4	400	0.25	33	39.7
C.125.9	125	0.1	34	37.81
C.250.9	250	0.1	44	56.24
DSJC125.1	125	0.09	34	38.4
DSJC125.5	125	0.5	10	11.47
DSJC125.9	125	0.9	4	4
gen200_p0.9_44	200	0.1	44	44
keller4	171	0.35	11	14.01
p_hat300-1	300	0.76	8	10.07
p_hat300-2	300	0.51	25	26.96
p_hat300-3	300	0.26	36	41.17
p_hat500-1	500	0.75	9	13.07
p_hat500-2	500	0.5	36	38.98
p_hat500-3	500	0.25	50	58.57
p_hat700-1	700	0.75	11	15.12
p_hat700-2	700	0.5	44	49.02
p_hat700-3	700	0.25	62	72.74
san400_0.5-1	400	0.5	13	13
san400_0.7-1	400	0.3	40	40
san400_0.7-2	400	0.3	30	30
san400_0.7-3	400	0.3	22	22
san400_0.9-1	400	0.1	100	100
sanr200_0.7	200	0.3	18	23.84
sanr200_0.9	200	0.1	42	49.27

Table 1: Graph characteristics

1. UB_{E} achieved by formulation (18) in which the quadratic constraint is handled by a second-order conic reformulation and the relaxation is solved by the Gurobi barrier algorithm;

2. UB_{BTN} corresponding to the optimal value of the Ben-Tal and Nemirovski reformulation; the parameter d has been fixed to $d = 8$ for

all graphs, as a result of a preliminary investigation.

3. the value UB_{K} returned by the Kelley cutting-plane algorithm after 300 iterations (the algorithm terminates early if the target ellipsoid bound is achieved);

4. the value UB_{C} returned by Algorithm 1 after a fixed number of iterations (from 5 to 50, algorithm terminates early if the target ellipsoid bound is achieved). The separation heuristic for clique inequalities is executed "aggressively" (i.e., repeatedly invoked until no further clique violated are found) if the graph density $\leq 25\%$, but "moderately" (i.e., just one call for each Kelley cut) if the graph density is $> 25\%$.

Columns %gap($\theta(G)$) reports the percentage gap $\frac{UB-\theta(G)}{\theta(G)}\%$ w.r.t. the Lovász θ bound. Notice that the approximation error is small for most of the instances, with a few exceptions. The most remarkable ones correspond to the graphs `p_hat300-1`, `p_hat500-1` and `p_hat700-1`. However, even in these cases, the linearization of the ellipsoidal constraint will provide a practical benefit.

Table 2 also shows that the quality of the upper bound UB_{E} is preserved by all the linearization techniques. In fact, the approximations achieved by UB_{BTN} and even by UB_{C} are satisfactory. This was somehow expected from the BTN reformulation as it provides a guaranteed approximation. Interestingly, also the Kelley cutting plane method achieves a strong bound, but Algorithm 1 exhibits significantly better performance. This is clear from the computing times (in seconds) and formulation details reported in Table 3. Precisely, for each graph, it contains the following data: time required by the subgradient algorithm to compute the initial ellipsoid; time to solve the SOCP reformulation, number of variables and constraints of the BTN reformulation and the time elapsed by the Gurobi LP solver to solve it; number of Kelley cuts along with separation time and reoptimization time; number of iterations of Algorithm 1, number of clique cuts generated by the cutting plane along with the Kelley cuts separation time and reoptimization time (which includes the additional clique inequalities separation).

The efficacy of Algorithm 1 is witnessed by the fact that a small number of Kelley cuts is sufficient to achieve good upper bounds. We remark that much worse bounds are attained without such cuts, even if we include in the formulation all the clique inequalities generated during the cutting-plane reoptimization. The Kelley cuts provide a significant formulation strengthening, yielding two additional benefits.

The first benefit concerns dense graphs, in which the standard formulation size may become intractable. In fact, as the density goes over $40-50\%$, the number of clique inequalities grows rapidly and overloads the simplex reoptimization. However, by applying the Kelley cuts, many of the clique

	Ellipsoid		BTN reformulation		Kelley's cuts		Kelley's and clique cuts	
Graph	UB_{E}	%gap($\theta(G)$)	UB_{BTN}	%gap($\theta(G)$)	UB_{K}	%gap($\theta(G)$)	UB_{C}	%gap($\theta(G)$)
brock200_1	27.46	0.00	27.46	0.00	28.45	3.61	28.53	3.90
brock200_2	14.53	2.11	14.93	4.92	15.25	7.17	15.16	6.54
brock200_3	19.09	1.43	19.44	3.29	19.60	4.14	19.60	4.14
brock200_4	21.53	1.13	21.81	2.44	22.26	4.56	22.35	4.98
brock400_1	40.35	1.64	40.72	2.57	42.11	6.07	41.83	5.37
brock400_2	41.71	5.43	42.38	7.13	43.22	9.25	42.82	8.24
brock400_3	40.25	1.95	40.34	2.18	42.11	6.66	41.05	3.98
brock400_4	40.86	2.92	40.91	3.05	41.45	4.41	41.26	3.93
C.125.9	37.99	0.49	38.16	0.94	38.29	1.28	38.48	1.79
C.250.9	57.02	1.39	57.43	2.12	58.27	3.61	58.09	3.29
DSJC125.1	38.54	0.37	38.75	0.92	38.84	1.15	39.16	1.99
DSJC125.5	11.82	3.05	11.87	3.49	11.91	3.84	12.24	6.71
DSJC125.9	4.12	3.00	4.13	3.25	4.28	7.00	4.31	7.75
gen200_p0.9_44	44.38	0.86	44.66	1.50	44.66	1.50	44.38	0.86
keller4	14.28	1.93	14.43	3.00	14.28	1.93	14.55	3.85
p_hat300-1	10.90	8.24	10.95	8.74	12.11	20.26	11.13	10.53
p_hat300-2	27.25	1.08	27.83	3.23	28.10	4.23	28.50	5.71
p_hat300-3	41.69	1.26	41.73	1.36	43.07	4.62	43.02	4.49
p_hat500-1	14.53	11.17	14.92	14.15	16.47	26.01	15.92	21.81
p_hat500-2	39.53	1.41	40.43	3.72	40.85	4.80	41.02	5.23
p_hat500-3	59.97	2.39	60.40	3.12	61.86	5.62	61.79	5.50
p_hat700-1	18.27	20.83	18.29	20.97	19.91	31.68	19.39	28.24
p_hat700-2	51.17	3.39	51.54	5.14	52.05	6.18	52.05	6.18
p_hat700-3	74.09	1.86	75.27	3.48	77.36	6.35	76.34	4.95
san400_0.5-1	13.41	3.15	13.42	3.23	13.28	2.15	13.46	3.54
san400_0.7-1	40.02	0.05	40.12	0.30	40.02	0.05	40.03	0.08
san400_0.7-2	30.04	0.13	30.09	0.30	30.01	0.03	30.04	0.13
san400_0.7-3	22.66	3.00	22.66	3.00	22.95	4.32	22.93	4.23
san400_0.9-1	100.00	0.00	100.24	0.24	100.00	0.00	100.00	0.00
sanr200_0.7	23.95	0.46	24.44	2.52	24.71	3.65	24.75	3.82
sanr200_0.9	49.64	0.75	49.85	1.18	50.33	2.15	50.60	2.70

Table 2: Ellipsoid vs. LP upper bounds

	Ellipsoid		BTN reformulation			Kelley's cuts		Kelley's and clique cuts		
Graph	Evaluation time	Barrier time	Columns	Rows	LP time	Kelley cuts	LP time	Kelley cuts	Cliques cuts	LP time
brock200_1	8.30	1.20	3,801	6,839	2.89	300	27.67	40	1,996	9.88
brock200_2	6.23	0.91	3,801	6,624	7.73	300	22.98	5	4,050	5.80
brock200_3	19.47	1.46	3,801	6,828	7.02	300	28.79	30	3,437	14.37
brock200_4	8.49	1.54	3,801	6,870	7.60	300	20.45	30	5,550	20.12
brock400_1	74.41	40.28	7,601	16,051	49.88	300	408.34	40	11,432	91.91
brock400_2	45.35	38.81	7,601	16,095	46.92	300	474.53	40	5,877	120.07
brock400_3	34.45	40.82	7,601	16,095	83.59	300	360.51	40	5,440	84.11
brock400_4	80.71	40.25	7,601	16,033	85.35	300	297.53	40	5,829	88.68
C.125.9	3.45	0.08	2,376	3,733	0.26	300	5.91	50	38	0.60
C.250.9	45.36	1.07	4,751	8,132	4.90	300	56.44	50	325	5.02
DSJC125.1	6.40	0.09	2,376	3,708	0.24	300	4.94	50	19	0.44
DSJC125.5	4.24	0.19	2,376	3,895	0.47	300	4.93	5	1,916	2.27
DSJC125.9	3.17	0.03	2,376	3,368	0.21	300	1.17	5	1,073	2.16
gen200_p0.9_44	20.48	0.36	3,801	6,212	0.87	2	0.01	2	25	0.07
keller4	3.99	0.20	3,250	4,959	0.58	300	7.34	10	839	0.64
p_hat300-1	29.92	0.87	5,701	8,926	2.72	300	43.83	5	8,424	4.23
p_hat300-2	39.97	2.77	5,701	9,819	21.44	300	81.99	5	1,837	3.35
p_hat300-3	85.42	6.00	5,701	10,949	26.37	300	73.98	40	2,537	25.94
p_hat500-1	166.87	6.84	9,501	15,709	54.74	300	128.01	5	23,585	40.46
p_hat500-2	241.38	24.59	9,501	17,805	193.14	300	389.02	5	12,185	93.69
p_hat500-3	211.57	88.85	9,501	20,808	305.52	300	404.79	50	6,082	168.52
p_hat700-1	372.69	28.42	13,301	22,768	239.73	300	1,392.81	5	43,855	215.79
p_hat700-2	479.25	123.83	13,301	26,270	338.15	300	3,344.90	10	18,215	230.92
p_hat700-3	1,427.96	602.93	13,301	32,486	778.75	300	2,045.77	50	25,834	291.83
san400_0.5-1	12.75	2.73	7,601	12,155	7.90	300	43.93	1	6,774	23.14
san400_0.7-1	16.85	21.88	7,601	15,890	31.63	10	12.73	2	757	4.76
san400_0.7-2	20.82	19.44	7,601	14,782	26.23	300	42.52	10	2,723	17.84
san400_0.7-3	758.07	9.72	7,601	13,618	37.01	300	11.59	1	996	3.04
san400_0.9-1	4.29	16.49	7,601	15,445	24.27	1	0.52	1	53	0.27
sanr200_0.7	12.23	1.32	3,801	6,873	9.99	300	26.49	30	1,640	6.02
sanr200_0.9	28.35	0.50	3,801	6,332	3.91	300	25.03	30	125	1.42

Table 3: CPU times and other details

inequalities can be discarded from the current formulation, while keeping safe the quality of the upper bound, yielding an important speed-up.

A second benefit is that using a small number of Kelley cuts protects the method from a potential difficulty caused by the cut density. Kelley cuts are structurally quite dense, a feature that typically degrades the computational behaviour of cutting planes (see the CPU times of the pure Kelley cutting-plane algorithm). On the contrary, a selected bunch of Kelley cuts is cost-effective even when used in branch-and-cut frameworks, as illustrated in the next section.

The overall picture drawn by this experiment is that linearizing the ellipsoid constraint yields strong and computationally tractable LP relaxations of stable set problems with size $|V|$ in the range $[300, 700]$.

We remark that lift-and-project methods have been previously investigated [15, 14] with the purpose of achieving target bounds comparable to those from SDP formulations. Compared to those, the method based on ellipsoids shows a more robust and density-independent behavior. In fact, dense graphs with remarkably larger size could be tackled than those which were tractable by lift-and-project (which, up to now, has been successfully applied to graphs with more than 300 vertices only if they are quite sparse). Interestingly, (building and) solving the BTN and Kelley's reformulations seems to be faster than solving strong SDP relaxations, such as those investigated in [7], [9] and [19].

4.5 Branch-and-cut

In the light of the previous experiments, we started looking at exact algorithms. Preliminary computations showed that using the BTN reformulation is not straightforward, due to its size. On the contrary, following the line of the previous section, a natural choice is to design a branch-and-cut algorithm based on Kelley's cutting planes with additional clique inequalities. In order to have a clear indication about the practical impact of these cuts, we tested a straightforward branch-and-cut implementation. Specifically, we load into an initial pool all the Kelley's cuts along with the clique inequalities generated by Algorithm 1 and then run the Gurobi branch-and-cut algorithm with checking for violated cuts in the pool. All other cutting planes are turned off, as these turn out to be not effective.

Table 4.5 compares three algorithms: Gurobi (default settings) applied only to the linear part of formulation (18); Gurobi (default settings but `MIQCPMethod`=1) applied to the MIQCP formulation (18); the above branch-and-cut algorithm.

It is worth mentioning that Gurobi implements sophisticated strategies to handle convex quadratic inequalities: constraint reformulation from quadratic constraint to *second-order conic* constraint, linearization of quadratic terms on binary variables and outer approximation algorithm. The param-

eter `MIQCPMethod`=1 controls the method used to solve MIQCP models and value 1 corresponds to a linearized, outer-approximation approach (instead of solving continuous QCP relaxations at each node). In our experiments, this value achieved the best results.

For each algorithm we report the number of evaluated subproblems and the CPU time (this includes the time to compute the ellipsoid and to generate the Kelley's cuts). The speed-up factor of Kelley's based branch-and-cut w.r.t. the best competitor is also quoted.

The results show that leaving Gurobi to handle internally the "raw" quadratic constraint is in general not helpful. As one may expect, in most cases it considerably reduces the size of the enumeration tree, but it takes a longer time due to a high computational burden at each subproblem. In fact, this method fails to solve the problem within the time limit as the size gets large (`C.250.9`, `brock400` and `p_hat500`, `p_hat700`). On the opposite side, the initial "light" formulation solves more quickly than any strengthened version in 12 over 25 cases. However, note that this occurs mainly for relatively easy instances. Indeed, when dealing with harder instances, the impact of the Kelley cuts is significant. Specifically, they still help to reduce the search tree, but they also guarantee a nice speed-up in 12 (hard) cases.

The overall indication of this experiment is that the proposed Kelley cuts provide a linear reformulation of the quadratic constraint which turns out to outperform the advanced reformulation technique embedded in Gurobi. Notice that this happens despite the fact that our implementation is quite basic, as cutting planes are generated using only the initial ellipsoid (i.e., the one related to the root relaxation), and leaves the cut management to Gurobi. An advanced implementation could be devised by generating ellipsoids at each subproblem and dealing with related cut-selection issues.

5 Concluding Remarks

The key idea in this paper is that one can use an approximate dual solution to the standard SDP relaxation of the SSP to construct an ellipsoid that wraps reasonably tightly around the stable set polytope, and that this ellipsoid can be used to construct quite strong cutting planes in the original (linear) space. The computational results, though preliminary, indicate that this approach is promising.

There is, however, room for further improvement. In particular:

- Instead of using the 'optimal' ellipsoid to generate cutting planes, one could use some other ellipsoid, or indeed a whole family of ellipsoids.
- As mentioned in Subsection 2.2, the SDP relaxation (7)-(10) can be strengthened by adding valid inequalities. (For example, Schrijver [37] suggested adding the inequalities $X_{ij} \geq 0$ for all $\{i, j\} \notin E$.) Let

$\tilde{\theta}(G) < \theta(G)$ be an improved upper bound obtained by solving such a strengthened SDP. The proof of Theorem 3 can be modified to show the existence of an ellipsoid, say $\tilde{E}$, such that

$$\tilde{\theta}(G) = \max \left\{ \sum_{i \in V} x_i : x \in \tilde{E} \right\}.$$

Such an ellipsoid might produce stronger cutting planes. (The extreme points of STAB(G) would no longer be guaranteed to lie on the boundary of the ellipsoid, but this may not matter.)

- In a branch-and-cut context, one could perhaps re-optimise the dual SDP solution (and therefore the corresponding ellipsoid) after branching. This could yield stronger cuts at the non-root nodes of the enumeration tree. Of course, one would not wish to solve an SDP to do this. Instead, one could perhaps perform a few iterations of the subgradient method that we described in [17].

Finally, one could explore the possibility of adapting the ellipsoidal approach to other combinatorial optimisation problems. In our view, this is likely to work well only for problems that have a 'natural' formulation as a continuous optimisation problem with a linear objective function and non-convex quadratic constraints, like the formulation (4)–(6) of the SSP.

Acknowledgment:

The second author was partially supported by the Engineering and Physical Sciences Research Council under grant `EP/D072662/1`.

References

[1] F. Alizadeh & D. Goldfarb (2003) Second-order cone programming. *Math. Program.*, 95, 3–51.

[2] Alpen-Adria-Universität Klagenfurt website `http://www.math.uni-klu.ac.at/or/Software`.

[3] K. Andersen & Y. Pochet (2009) Coefficient strengthening: a tool for reformulating mixed-integer programs. *Math. Program.*, 122, 121–154.

[4] R. Borndörfer (1998) *Aspects of Set Packing, Partitioning and Covering*. Doctoral Thesis, Technical University of Berlin.

[5] E. Balas, S. Ceria, G. Cornuéjols & G. Pataki (1996) Polyhedral methods for the maximum clique problem. In D.S. Johnson & M.A. Trick (eds.), *Cliques, Coloring and Satisfiability*. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 26, pp. 11–28.

[6] A. Ben-Tal & A. Nemirovski (2001) On polyhedral approximations of the second order cone. *Math. Oper. Res.*, 26, 193–205.

[7] Burer, S., Vandenbussche, D.: Solving lift-and-project relaxations of binary integer programs. SIAM J. on Opt., 16, 726–750 (2006).

[8] DIMACS Repository
`ftp://dimacs.rutgers.edu/pub/challenge/graph/benchmarks/clique`.

[9] I. Dukanovic & F. Rendl (2007) Semidefinite programming relaxations for graph coloring and maximal clique problems. *Math. Program.*, 109, 345–365.

[10] T. Fahle (2002) Simple and fast: Improving a branch-and-bound algorithm for maximum clique. *Proc. 10th European Symposium on Algorithms*, Lecture Notes in Computer Science vol. 2461, Springer-Verlag, pp. 47-86.

[11] T. Fujie & M. Kojima (1997) Semidefinite programming relaxation for nonconvex quadratic programs. *J. Glob. Opt.*, 10, 367–380.

[12] T. Fujie & A. Tamura (2002) On Grötschel-Lovász-Schrijver's relaxation of stable set polytopes. *J. Oper. Res. Soc. Japan*, 45, 285–292.

[13] M. Giandomenico & A.N. Letchford (2006) Exploring the relationship between max-cut and stable set relaxations. *Math. Program.*, 106, 159–175.

[14] M. Giandomenico, F. Rossi & S. Smriglio (2013) Strong lift-and-project cutting planes for the stable set problem. *Math. Program.*, 141, 165–192.

[15] M. Giandomenico, A. Letchford, F. Rossi & S. Smriglio (2009) An application of the Lovász-Schrijver $M(K, K)$ operator to the stable set problem. *Math. Program.*, 120, 381–401.

[16] M. Giandomenico, A.N. Letchford, F. Rossi & S. Smriglio (2011) A new approach to the stable set problem based on ellipsoids. In O. Günlük & G.J. Woeginger (eds.) *Integer Programming and Combinatorial Optimization XV*. Lecture Notes in Computer Science, vol. 6655. Heidelberg: Springer.

[17] M. Giandomenico, A.N. Letchford, F. Rossi & S. Smriglio (2013) Approximating the Lovász theta function with the subgradient method. *Elec. Notes in Discr. Math.*, 41, 157–164.

[18] M. Grötschel, L. Lovász & A.J. Schrijver (1988) *Geometric Algorithms in Combinatorial Optimization*. New York: Wiley.

[19] G. Gruber & F. Rendl (2003) Computational experience with stable set relaxations. *SIAM J. Opt.*, 13, 1014–1028.

[20] D.S. Johnson & M.A. Trick (eds.) *Cliques, Coloring and Satisfiability: the 2nd DIMACS Implementation Challenge*. American Mathematical Society, Providence, RI.

[21] J. Håstad (1999) Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Math.*, 182, 105–142.

[22] R.M. Karp (1972) Reducibility among combinatorial problems. In R.E. Miller & J.W. Thatcher (eds) *Complexity of Computer Computations*, New York, Plenum, pp. 85–103.

[23] J.E. Kelley (1960) The cutting-plane method for solving convex programs. *SIAM Journal*, 8, 703–713.

[24] C. Lemaréchal & F. Oustry (2001) SDP relaxations in combinatorial optimization from a Lagrangian viewpoint. In N. Hadjisawas & P.M. Pardalos (eds.), *Advances in Convex Analysis and Global Optimization*. Dortrecht: Kluwer.

[25] M. Lobo, L. Vandenberghe, S. Boyd & H. Lebret (1998) Applications of second-order cone programming. *Lin. Alg. App.*, 284, 193–228.

[26] L. Lovász (1979) On the Shannon capacity of a graph. *IEEE Trans. Inform. Th.*, IT-25, 1–7.

[27] L. Lovász & A.J. Schrijver (1991) Cones of matrices and set-functions and 0-1 optimization. *SIAM J. Optimization*, 1, 166–190.

[28] C.J. Luz & A.J. Schrijver (2005) A convex quadratic characterization of the Lovász theta number. *SIAM J. Discr. Math.*, 19, 382–387.

[29] J. Malick, J. Povh, F. Rendl, A. Wiegele (2007) *Boundary Point Method for solving SDPs:* `mprw.m`, Inst. f. Mathematik, Alpen-Adria-Universität Klagenfurt.

[30] G.L. Nemhauser & G. Sigismondi (1992) A strong cutting plane/branch-and-bound algorithm for node packing. *J. Oper. Res. Soc.*, 43, 443–457.

[31] G.L. Nemhauser & L.A. Wolsey (1988) *Integer and Combinatorial Optimization*. New York: Wiley.

[32] M.W. Padberg (1973) On the facial structure of set packing polyhedra. *Math. Program.*, 5, 199–215.

[33] S. Poljak, F. Rendl & H. Wolkowicz (1995) A recipe for semidefinite relaxation for (0,1)-quadratic programming. *J. Global Opt.*, 7, 51–73.

[34] S. Rebennack, M. Oswald, D.O. Theis, H. Seitz, G. Reinelt & P.M. Pardalos (2011) A branch and cut solver for the maximum stable set problem. *J. Comb. Opt.*, 21, 434–457.

[35] J.-C. Régin (2003) Solving the maximum clique problem with constraint programming. In *Proceedings of CPAIOR'03*, Lecture Notes in Computer Science vol. 2883, Springer, pp. 634–648.

[36] F. Rossi & S. Smriglio (2001) A branch-and-cut algorithm for the maximum cardinality stable set problem. *Oper. Res. Lett.*, 28, 63–74.

[37] A.J. Schrijver (1979) A comparison of the Delsarte and Lovász bounds. *IEEE Trans. Inf. Th.*, IT-25, 425–429.

[38] Tunçel (2001) On the Slater condition for SDP relaxations of nonconvex sets. *Oper. Res. Lett.*, 29, 181–186.

	Gurobi linear formulation		Gurobi MIQCP formulation		Branch-and-cut		
Graph name	Subproblems	Time	Subproblems	Time	Subproblems	Time	**Speed-up**
brock200_1	283,763	429.23	**88,176**	419.81	135,791	**251.701**	1.71
brock200_2	5,600	33.76	4,469	105.20	**1,764**	**33.08**	1.02
brock200_3	18,890	**48.84**	**9,794**	172.28	15,306	82.506	—
brock200_4	35,940	**74.64**	**15,619**	179.35	**15,619**	101.36	—
brock400_4	3,926,519	51,644.22	***	***	**1,676,237**	**46,730.35**	1.11
C.125.9	3,940	**4.98**	**2,790**	15.29	3,636	7.47	—
C.250.9	53,842,560	65,064.41	***	***	**29,525,099**	**52,996.94**	1.23
DSJC125.1	3,599	**3.52**	**2,983**	21.18	3,940	9.23	—
DSJC125.5	440	**4.22**	**258**	8.28	342	7.59	—
DSJC125.9	**0**	**0.37**	**0**	3.49	**0**	5.54	—
gen200_p0.9_44	1,132	**6.02**	627	23.14	**48**	22.45	—
keller4	4,484	**9.05**	6,009	13.72	**3,745**	23.35	—
p_hat300-1	5,818	179.23	**5,135**	134.94	5,675	**133.8**	1.34
p_hat300-2	6,123	286.60	**3,689**	356.51	4,933	**164.08**	1.75
p_hat300-3	1,498,311	7,650.19	**225,345**	4,491.96	558,861	**3,305.63**	2.31
p_hat500-1	67,086	1,418.18	***	***	**59,089**	**699.94**	2.03
p_hat500-2	268,865	5,266.73	***	***	**173,621**	**4,510.07**	1.17
p_hat700-1	168,289	7,118.61	***	***	**147,494**	**3,561.526**	2.00
san400_0.5-1	29	**14.77**	31	26.17	**0**	39.64	—
san400_0.7-1	**0**	**11.69**	**0**	21.42	**0**	24.48	—
san400_0.7-2	501	55.15	**0**	**37.26**	1	41.33	1.33
san400_0.7-3	**0**	**2.72**	**0**	764.53	**0**	763.65	—
san400_0.9-1	**0**	**0.98**	**0**	5.48	**0**	5.33	—
sanr200_0.7	118,261	242.11	**44,989**	241.63	73,687	**136.54**	1.77
sanr200_0.9	1,171,075	921.88	**243,331**	1,454.59	302,031	**303.15**	3.04

Table 4: Branch-and-cut results