Ellipsoidal Relaxations of the Stable Set Problem: Theory and Algorithms

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

To appear in *SIAM Journal on Optimization*

Abstract

A new exact approach to the stable set problem is presented, which attempts to avoid 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, 29, 33, 34, 35]). Moreover, it is a remarkable fact that sophisticated mathematical programming algorithms for the SSP, such as those in [14, 29, 33, 35], have not performed significantly better than relatively simple algorithms based on implicit enumeration, such as those in [10, 34].

[*]Dipartimento di Informática, Università Di L'Aquila, Via Vetoio, 67010, Coppito (AQ), Italy. E-mail: `monia.giandomenico@gmail.com, fabrizio.rossi, stefano.smrigliog@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. Section 4 discusses how to use the ellipsoids within exact algorithms for the SSP. Some computational results are presented in Section 5, and concluding remarks are made in Section 6.

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., [3, 18, 31]). The most well-known facet-defining inequalities for STAB(G) are the *clique* inequalities of Padberg [31]. 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:

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

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 [30]. On the other hand, some fast and effective separation heuristics exist, not only for clique inequalities, but also for various other inequalities (e.g., [3, 14, 29, 33, 35]). 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 [25] 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:

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

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:

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

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

Lovász [25] 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., [4, 9, 19, 33]). 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., [3, 9, 13, 19, 26, 36]). 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:

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

Then we have:

$$\text{STAB}(G) \subseteq \text{TH}(G) \subseteq \text{QSTAB}(G).$$

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

The set $\text{TH}(G)$ is called the *theta body*. A characterisation of $\text{TH}(G)$ 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 $\operatorname{Diag}(\lambda) + M(\mu)$ is psd, the set

$$E(\lambda, \mu) = \left\{ x \in \mathbb{R}^n : \, x^T (\operatorname{Diag}(\lambda) + M(\mu)) x \leq \lambda \cdot x \right\} \qquad (11)$$

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

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

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

2.4 Relaxations of non-convex quadratic problems

For what follows, we will also need to briefly review Lagrangian and SDP relaxations of general non-convex quadratic problems. Given a problem of the form:

$$\begin{array}{rl} \inf & x^T Q^0 x + c^0 \cdot x \\ \text{s.t.} & x^T Q^j x + c^j \cdot x = b_j \quad (j = 1, \dots, r) \\ & x \in \mathbb{R}^n, \end{array} \tag{12}$$

Shor [37] proposed to relax the constraints (12) in Lagrangian fashion, 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:

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

Shor also introduced the following SDP, which we call the *dual* SDP:

$$\begin{array}{rl} & \sup \; t \\ \text{s.t.} & \begin{pmatrix} -t - \phi \cdot b & \left(c^0 + \sum_{j=1}^r \phi_j c^j\right)^T / 2 \\ \left(c^0 + \sum_{j=1}^r \phi_j c^j\right) / 2 & Q^0 + \sum_{j=1}^r \phi_j Q^j \end{pmatrix} \succeq 0 \\ & \phi \in \mathbb{R}^r, \; t \in \mathbb{R}. \end{array}$$

He showed:

Theorem 2 (Shor [37]) *The dual SDP yields the same lower bound as the Lagrangian dual. Moreover, a vector ϕ^* is an optimal solution to the Lagrangian dual if and only if there exists an optimal solution (ϕ^*, t^*) to the dual SDP.*

Note that, if such a vector ϕ^* exists, then the function $f(x, \phi^*)$ is convex, since the matrix $Q^0 + \sum_{j=1}^r \phi_j^* Q^j$ is psd.

As noted in [11, 32], Shor's SDP is the dual of the following more natural SDP relaxation of the original problem, which we call the *primal* SDP:

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

where $Q^j \bullet X$ denotes $\sum_{i=1}^n \sum_{k=1}^n Q^j_{ik} X_{ik}$. Thus, if either or both of the primal and dual SDP satisfy Slater's condition, then the primal SDP yields the same lower bound as the Lagrangian dual.

In our recent paper [17], these results are applied to the SSP, yielding a subgradient algorithm that approximates $\theta(G)$ from above. Luz & Schrijver [27] 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). In this section, we investigate the ellipsoids in more detail. In Subsection 3.1, we show how to construct ellipsoids that yield strong upper bounds on $\alpha(G)$. In Subsection 3.2, we consider two other desirable properties for ellipsoids to have. In Subsection 3.3, we show that, perhaps surprisingly, the best ellipsoid according to the first criterion can be bad according to the other two criteria. This analysis is based on a special class of graphs, including the perfect graphs.

We note in passing that every point $x \in \mathbb{R}^n$ satisfying (5), (6) also satisfies at equality the constraint in (11). This implies that every extreme point of $\mathrm{STAB}(G)$ lies on the boundary of every ellipsoid.

3.1 Optimal ellipsoids

Recall that Theorem 1 expresses $\mathrm{TH}(G)$ as the intersection of the entire infinite family of ellipsoids. We will show that there exists at least one ellipsoid with a very desirable property. First, however, we need the dual of the SDP (7)–(10), which can be written in the form:

$$\min\ t \tag{13}$$

$$\text{s.t.} \quad \begin{pmatrix} t & -(e+\lambda)^T/2 \\ -(e+\lambda)/2 & \mathrm{Diag}(\lambda) + M(\mu) \end{pmatrix} \succeq 0 \tag{14}$$

$$\lambda \in \mathbb{R}^n,\ \mu \in \mathbb{R}^m,\ t \in \mathbb{R}, \tag{15}$$

where e is the vector of all ones. Here, λ and μ are the vectors of dual variables for the constraints (8) and (9), respectively. (As before, t is the dual variable for the constraint $Y_{00} = 1$.)

We are now ready to present our main result:

Theorem 3 *There exists at least one optimal solution to the dual SDP (13)-(15). Moreover, for any such solution (λ^*, μ^*, t^*), we have:*

$$\theta(G) = t^* = \max\left\{\sum_{i\in V} x_i : x \in E(\lambda^*, \mu^*)\right\}. \tag{16}$$

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 solution (λ^*, μ^*, t^*) exists, strong duality holds, and $t^* = \theta(G)$.

Now consider a Lagrangian relaxation of the non-convex quadratic problem (4)–(6), in which all constraints are relaxed. This relaxation can be written as:

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

where $\tilde{\lambda} \in \mathbb{R}^n$ and $\tilde{\mu} \in \mathbb{R}^m$ are the vectors of Lagrangian multipliers for the constraints (5) and (6), respectively. Applying Theorem 2 in Subsection 2.4, but switching signs to take into account the fact that the objective (7) is of maximisation type, an optimal solution to the Lagrangian dual is obtained by setting $\tilde{\lambda}$ to λ^* and $\tilde{\mu}$ to μ^*, and the optimal value of the Lagrangian dual is equal to $\theta(G)$. In other words, $\theta(G)$ is equal to:

$$\max\left\{\sum_{i\in V} x_i - x^T(\mathrm{Diag}(\lambda^*) + M(\mu^*))x + \lambda^*\cdot x : \ x \in \mathbb{R}^n\right\}. \tag{17}$$

Moreover, the matrix $\mathrm{Diag}(\lambda^*) + M(\mu^*)$ is psd from (14), and therefore the objective function in (17) is concave.

Now, let us write the relaxation on the right-hand side of (16) as:

$$\max\left\{\sum_{i\in V} x_i : \ x^T(\mathrm{Diag}(\lambda^*) + M(\mu^*))x \le \lambda^*\cdot x, \ x \in \mathbb{R}^n\right\}. \tag{18}$$

We relax this further by moving the (one and only) constraint to the objective function, with non-negative Lagrangian multiplier ϕ. The result is:

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

Note that this reduces to (17) when ϕ is 1. Therefore, the upper bound from (18) must be at least as strong as the one from (17), which as we saw is equal to $\theta(G)$.

To complete the proof, it suffices to show that the optimal value of ϕ, which we denote by ϕ^*, is 1. Suppose that it was not equal to 1. Then we could obtain an upper bound that is better than $\theta(G)$ by taking (17) and multiplying λ^* and μ^* by ϕ^*. But this contradicts the fact that (λ^*, μ^*) is an optimal solution to the Lagrangian dual for (4)–(6). $\square$

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 $\theta(G)$. (We remark that a similar proof technique was used in [2] to reformulate 0-1 quadratic programs. The main difference between their method and ours, apart from the fact that they apply to different problems, is that they perturb the objective function, whereas we generate a valid convex quadratic constraint.)

We say that an ellipsoid $E(\lambda^*, \mu^*)$ is *optimal* if there exists a t^* such that (λ^*, μ^*, t^*) is an optimal solution to the dual of the SDP (or, equivalently, if (λ^*, μ^*) is an optimal solution to the Lagrangian dual). We illustrate this concept on three 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:

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

This can be written in the alternative form

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

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, $\alpha(G) = 2$ [31] and $\theta(G) = \sqrt{5}$ [18]. The unique optimal dual solution has $\lambda^*_i = 1$ for all $i \in V$ and $\mu^*_e = \sqrt{5} - 1$ for all $e \in E$. The corresponding convex quadratic constraint is:

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

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 "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".

Definition 2 *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 (pd).*

In principle, instead of using a weak ellipsoid to construct a relaxation of STAB(G), we could take the intersection of two or more strong ellipsoids to obtain a tighter relaxation. Unfortunately, we are not aware of any polynomial-time algorithm to test if a given ellipsoid $E(\lambda, \mu)$ is strong.

As for degenerate ellipsoids, we will see in Section 4 that they could cause numerical problems for our algorithms. Fortunately, we found that, for graphs of practical interest, optimal ellipsoids are very rarely degenerate. (One can easily check if $E(\lambda, \mu)$ is degenerate, since one can easily check whether a rational matrix is psd or pd [18].)

To make these concepts more concrete, we apply them to 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:

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

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. $\square$

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 $0 \le \sum_{i \in V} x_i \le 1$. So, the ellipsoid is degenerate. On the other hand, the inequality $\sum_{i \in V} x_i \le 1$ is a clique inequality, which defines a facet of STAB(G). Since no other ellipsoid has this facet on its boundary, the optimal ellipsoid is strong. $\square$

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 $\mu_e^* = \sqrt{5} - 1$ for all $e \in E$. The minimum eigenvalue of the associated matrix $\mathrm{Diag}(\lambda) + M(\mu)$ 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. $\square$

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:

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

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:

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

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

3.3 Ellipsoids, clique covers 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, $\bar{\chi}^f(G) \le \bar{\chi}(G)$, and we already saw in Subsection 2.2 that $\alpha(G) \le \theta(G) \le \bar{\chi}^f(G)$. A graph G is called *perfect* if $\alpha(G') = \bar{\chi}(G')$ holds for every node-induced subgraph $G' \subseteq G$. The following proposition states 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. 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. When G is perfect, $\mathrm{STAB}(G) = \mathrm{QSTAB}(G)$, and is therefore completely defined by clique and non-negativity inequalities. $\square$

Next, consider the class of graphs for which $\alpha(G) = \bar{\chi}(G)$. This includes all perfect graphs of course, but it includes many other graphs besides. (For example, let G be the graph obtained from the 5-hole by adding a new node 6 and a new edge $\{1,6\}$. Then $\alpha(G) = \bar{\chi}(G) = 3$, yet G is not perfect.) The following proposition shows that, for the graphs in this class, there exist optimal ellipsoids of a particularly simple form.

Proposition 2 *Suppose that $\alpha(G) = \bar{\chi}(G)$. Let $C_1, \ldots, C_{\alpha(G)} \subset V$ be any family of cliques in G 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*

$$\sum_{k=1}^{\alpha(G)} \left(\sum_{i \in C'_k} x_i \right)^2 \leq \sum_{k=1}^{\alpha(G)} \sum_{i \in C'_k} x_i \tag{19}$$

defines an optimal ellipsoid.

Proof. First, observe that the inequality (19) can be obtained from (11) by setting λ_i to 1 for all $i \in V$ and setting μ_{ij} to 2 if $\{i,j\} \subseteq C'_k$ for some k, and to 0 otherwise. Moreover, the left-hand side of (19) is a sum of squares, and therefore the inequality is convex, as stated.

Now, let $f(x)$ denote the left-hand side of (19) minus the right-hand side, so that the specified ellipsoid is defined by the constraint $f(x) \leq 0$. Also let

$x^* \in \mathbb{R}^n$ be an arbitrary point such that $\sum_{i \in C'_k} x^*_i = 1$ for $k = 1, \ldots, \alpha(G)$. Note that $f(x^*) = 0$, and therefore x^* is on the boundary of the ellipsoid. A first-order Taylor series expansion of $f(x)$ at x^* shows that the tangent hyperplane to the ellipsoid at x^* is given by the equation $\nabla f(x^*) \cdot (x - x^*) = 0$. Now, for $k = 1, \ldots, \alpha(G)$ and any $i \in C'_k$, we have $\partial f / \partial x_i = 2 \sum_{j \in C'_k} x_j - 1$, which takes the value 1 at x^*. So, the tangent hyperplane is defined by the equation $\sum_{i \in V} x_i - \sum_{i \in V} x^*_i = 0$. But $\sum_{i \in V} x^*_i = \alpha(G)$, and therefore the equation reduces to $\sum_{i \in V} x_i = \alpha(G)$. So, all points in the ellipsoid satisfy $\sum_{i \in V} x_i \leq \alpha(G)$. This implies that the given pair (λ, μ) is an optimal solution to the Lagrangian dual, and therefore that the ellipsoid is optimal. $\square$

Counter-intuitively, 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. Note that the matrix of quadratic terms on the left-hand side of (19) is block-diagonal, with $\alpha(G)$ blocks. This implies that its rank is $\alpha(G)$. Now, if G is a stable set, the matrix is of full rank (in fact, it is the identity matrix). It is therefore pd, and the ellipsoid is non-degenerate. If, on the other hand, G is not a stable set, then $\alpha(G) < n$, and the matrix is singular. It therefore cannot be pd, and the ellipsoid is degenerate.

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 (19) is one of them if and only if $\alpha(G) = 1$, i.e., if G is a clique. $\square$

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.

4.1 Computation of a near-optimal ellipsoid

In practice, the dual SDP can be solved only to limited precision. Therefore, we must be content with a "near-optimal" dual solution. It is however crucial for the dual solution to be feasible, i.e., for the matrix $\mathrm{Diag}(\lambda) + M(\mu)$ to be psd, since otherwise the resulting set $E(\lambda, \mu)$ would not be an ellipsoid. In the conference version of this paper [16], the dual solution was obtained using the SDP solver described in [28]. Afterwards, however, 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, for this paper we used our subgradient approach to compute the dual solution, which we denote by $(\bar{\lambda}, \bar{\mu})$.

4.2 An initial convex relaxation of the SSP

We start by constructing a collection $\mathcal{C}$ of cliques covering all the edges of G, using a standard greedy algorithm (see, e.g., [14]). Now, let $x^T Ax \leq b \cdot x$ be the convex quadratic constraint associated with the ellipsoid $E(\bar{\lambda}, \bar{\mu})$. Our initial convex relaxation of the SSP is:

$$\begin{array}{rl} \max & \sum_{i \in V} x_i \\ \text{s.t.} & x^T Ax \leq b \cdot x \\ & \sum_{i \in C} x_i \leq 1 \quad (C \in \mathcal{C}) \\ & x \in [0,1]^n. \end{array} \tag{20}$$

Note that this is a (convex) quadratically constrained program (QCP). In principle, this relaxation could be used directly within a plain branch-and-bound algorithm. We however use a more sophisticated approach, as we explain in the following subsections.

4.3 Alternative representations of the quadratic constraint

In practice, it is helpful to reformulate the quadratic constraint in (20). We found that three options worked well.

The first option is to convert the quadratic constraint into a *second-order conic* (SOC) constraint. This can be profitable since such constraints can be handled quite efficiently via interior-point algorithms (see, e.g., [1, 24]). 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:

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

Notice that all these new constraints are linear, apart from the last, which is a SOC constraint.

The second option is to approximate the SOC constraint in $\mathbb{R}^n$ by the well-known linear description of Ben-Tal and Nemirovski [5]. They define an extended formulation in the space $\mathbb{R}^{n+d}$ with $d+1$ inequalities, where d is a parameter. Reformulation accuracy is proportional to 2^{d+1} (see [5] for details). Our motivation for doing this is that one then needs to solve only LP relaxations 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. We use the cutting-plane method of Kelley [23]. Given a convex constraint of the form $f(x) \leq 0$, 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 show 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 $0 < \epsilon < 1$ such that the point
$$\tilde{x} = \epsilon x^* + (1 - \epsilon)\hat{x}$$
lies on the boundary of the ellipsoid. (This can be done by solving a quadratic equation in the single real variable ϵ.)

3. Generate the Kelley cutting plane corresponding to $\tilde{x}$, which takes the form $(2A\tilde{x} - b) \cdot x \leq (2A\tilde{x} - b) \cdot \tilde{x}$.

We will call the resulting cutting-plane algorithm "Algorithm 0". This approach has the advantage that the branch-and-cut algorithm needs to solve only LPs that involve the original x variables.

The outer approximation can be further 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.

Algorithm 1 Kelley 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 $\Gamma \rightarrow a \cdot x \leq b$
        round down $\lfloor a \rfloor \cdot x \leq \lfloor b \rfloor$
        $\bar{\mathcal{P}} \leftarrow \bar{\mathcal{P}} \cup \{\lfloor a \rfloor \cdot x \leq \lfloor b \rfloor\}$
        optimize over $\bar{\mathcal{P}}$, get $x^*$
        execute clique cutting plane routine that
        returns a collection of clique inequalities $\mathcal{C}$;
        $\bar{\mathcal{P}} \leftarrow \bar{\mathcal{P}} \cup \mathcal{C}$
    else
        STOP
    end if
end for
return $\bar{\mathcal{P}}$
```

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 this reason, and also for the sake of brevity, we do not consider this option further in this paper.

4.4 Branch-and-cut

Finally, we discuss branch-and-cut algorithms. In principle, one can embed the relaxation (20) directly within a branch-and-bound algorithm, and then add known valid inequalities for the SSP (such as clique inequalities) at each node of the enumeration tree. Unfortunately, reoptimising after branching is slow when using interior-point methods. For the same reason, a branch-and-cut algorithm based on SOCP relaxations is not viable either. Not only that, but we have found that the Ben-Tal and Nemirovski [5] relaxation does not work well in a branch-and-cut context either, due partly to the presence of additional variables and constraints, and partly to numerical difficulties.

Our preferred choice, therefore, is to use a branch-and-cut algorithm with LP relaxations in the space of the original x variables, in which Kelley cuts and/or clique inequalities can be generated at each node.

5 Computational Results

We now perform an experimental comparison of the different algorithmic options. Two major issues are addressed: (*i*) showing that the θ-bound is computationally accessible by linear descriptions and (*ii*) checking whether such descriptions can be of use within a branch-and-cut framework.

5.1 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 [7]. Among them, we do not consider the "easy" ones, that is, those for which the initial set of clique inequalities included in the relaxation (20) provides an LP relaxation yielding the integer optimum, or showing a negligible integrality gap. The instance features are reported in Table 1.

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.

5.2 Strength of the linear descriptions

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

1. UB_{E} achieved by the first formulation mentioned in Subsection 4.3, in which the quadratic constraint is converted into a SOC constraint 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. UB_{K} returned by Algorithm 0 with iteration limit set to 300 (the algorithm terminates early if the target ellipsoid bound is achieved);

4. UB_{C} returned by Algorithm 1 with iteration limit set to 50 (the 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 violations are found) if the graph density $\leq 25\%$, but "moderately" (i.e., just one call for each Kelley cut) if the graph density is $> 25\%$.

The chosen iteration limits avoid a significant tailing-off effect for all instances and yield, on average, a profitable trade-off between quality of the upper bound and efficiency. The columns headed "%gap($\theta(G)$)" report the

Name	$\lvert V\rvert$	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

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. Notable exceptions are the graphs `p_hat300-1`, `p_hat500-1` and `p_hat700-1`, representing the densest members of the `p_hat` class. To explain this fact, recall that the initial ellipsoid $E(\bar{\lambda}, \bar{\mu})$ is computed by the subgradient algorithm presented in [17].

When the graph is dense, the subgradient algorithm has to handle a large number of Lagrangian multipliers, which leads to very slow convergence. So, for those instances, the ellipsoid generated is quite far from being optimal. Happily, we will show in Section 4 that our branch-and-cut algorithm is effective on those very instances, even if the ellipsoids are sub-optimal.

Table 2 also shows that the quality of the upper bound UB_{E} is preserved by all the linearization techniques. Indeed, the approximations achieved by UB_{BTN} and even by UB_{K}, UB_{C} are satisfactory. This was somehow expected from the BTN reformulation as it provides a guaranteed approximation. Interestingly, also Algorithm 0 achieves a strong bound, but Algorithm 1 exhibits significantly better performance with respect to CPU time. This is clear from the details reported in Table 3, in which t_{K} and t_{C} denotes the total CPU time (sum of separation and LP reoptimization times) of Algorithm 0 and Algorithm 1 respectively. Table 3 contains also the following data for each graph: time required by the subgradient algorithm to compute the initial ellipsoid; time required by the barrier algorithm to solve the SOCP reformulation; number of variables and constraints of the BTN reformulation and time elapsed by the Gurobi LP solver to solve it; number of Kelley cuts generated by Algorithm 0, upper bound and number of Kelley cuts by Algorithm 0 at t_{C}; number of Kelley cuts and clique cuts generated by Algorithm 1. All times are expressed in seconds.

Looking at the bounds computed by Algorithm 0 at t_{C}, one can observe that Algorithm 1 shows a faster convergence. Indeed it computes better bounds in 21 out of 31 cases, typically corresponding to hard instances. In the remaining 10 cases, even if the bound from Algorithm 0 is slightly stronger, it is achieved by a much larger number of Kelley cuts.

The Kelley cuts provide 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 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 ellipsoidal constraint yields strong and computationally tractable LP relaxations of stable set problems with size $|V|$ in the range $[300, 700]$.

	Ellipsoid		**BTN reformulation**		**Algorithm 0**		**Algorithm 1**	
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**			**Algorithm 0**				**Algorithm 1**		
Graph	Subgradient time	Barrier time	Columns	Rows	LP time	t_K	Kelley cuts	Bound at t_C	Kelley cuts at t_C	t_C	Kelley cuts	Clique cuts
brock200_1	8.3	1.2	3,801	6,839	2.89	27.67	300	28.52	178	9.88	40	1,996
brock200_2	6.23	0.91	3,801	6,624	7.73	22.98	300	15.26	58	5.8	5	4,050
brock200_3	19.47	1.46	3,801	6,828	7.02	28.79	300	19.62	149	14.37	30	3,437
brock200_4	8.49	1.54	3,801	6,870	7.6	20.45	300	22.29	296	20.12	30	5,550
brock400_1	74.41	40.28	7,601	16,051	49.88	408.34	300	42.75	142	91.91	40	11,432
brock400_2	45.35	38.81	7,601	16,095	46.92	474.53	300	43.34	155	120.07	40	5,877
brock400_3	34.45	40.82	7,601	16,095	83.59	360.51	300	42.19	120	84.11	40	5,440
brock400_4	80.71	40.25	7,601	16,033	85.35	297.53	300	41.53	121	88.68	40	5,829
C.125.9	3.45	0.08	2,376	3,733	0.26	5.91	300	38.43	53	0.6	50	38
C.250.9	45.36	1.07	4,751	8,132	4.9	56.44	300	59.19	61	5.02	50	325
DSJC125.1	6.4	0.09	2,376	3,708	0.24	4.94	300	39.11	54	0.44	50	19
DSJC125.5	4.24	0.19	2,376	3,895	0.47	4.93	300	12.01	169	2.27	5	1,916
DSJC125.9	3.17	0.03	2,376	3,368	0.21	1.17	113	N/A	N/A	2.16	5	1,073
gen200_p0.9_44	20.48	0.36	3,801	6,212	0.87	0.01	2	N/A	N/A	0.07	2	25
keller4	3.99	0.2	3,250	4,959	0.58	7.34	300	14.39	59	0.64	10	839
p_hat300-1	29.92	0.87	5,701	8,926	2.72	43.83	300	12.19	39	4.23	5	8,424
p_hat300-2	39.97	2.77	5,701	9,819	21.44	81.99	300	28.36	27	3.35	5	1,837
p_hat300-3	85.42	6	5,701	10,949	26.37	73.98	300	43.4	178	25.94	40	2,537
p_hat500-1	166.87	6.84	9,501	15,709	54.74	128.01	300	16.36	159	40.46	5	23,585
p_hat500-2	241.38	24.59	9,501	17,805	193.14	389.02	300	41.01	150	93.69	5	12,185
p_hat500-3	211.57	88.85	9,501	20,808	305.52	404.79	300	62.01	228	168.52	50	6,082
p_hat700-1	372.69	28.42	13,301	22,768	239.73	1,392.81	300	19.97	149	215.79	5	43,855
p_hat700-2	479.25	123.83	13,301	26,270	338.15	3,344.90	300	52.25	157	230.92	10	18,215
p_hat700-3	1,427.96	602.93	13,301	32,486	778.75	2,045.77	300	77.81	198	291.83	50	25,834
san400_0.5-1	12.75	2.73	7,601	12,155	7.9	13.93	11	N/A	N/A	23.14	1	6,774
san400_0.7-1	16.85	21.88	7,601	15,890	31.63	12.73	10	40.18	6	4.76	2	757
san400_0.7-2	20.82	19.44	7,601	14,782	26.23	42.52	300	30.85	33	17.84	10	2,723
san400_0.7-3	758.07	9.72	7,601	13,618	37.01	11.59	300	23.83	10	3.04	1	996
san400_0.9-1	4.29	16.49	7,601	15,445	24.27	0.27	1	N/A	N/A	0.52	1	53
sanr200_0.7	12.23	1.32	3,801	6,873	9.99	26.49	300	25.37	99	6.02	30	1,640
sanr200_0.9	28.35	0.5	3,801	6,332	3.91	25.03	300	50.96	51	1.42	30	125

Table 3: CPU times and other details

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 reformulations seems to be faster than solving strong SDP relaxations, such as those investigated in [6], [9] and [19].

5.3 Branch-and-cut

We then tested a straightforward branch-and-cut implementation. Specifically, we load into an initial pool all the Kelley 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 compares the branch-and-cut algorithm described in the previous paragraph with two other algorithms. The first consists of solving, by the Gurobi branch-and-cut routine (default settings), the 0-1 LP formulation based on the initial collection $\mathcal{C}$ of clique inequalities (see Section 4.2). This is referred to as the *clique formulation* and reads as:

$$\begin{array}{rll} \max & \sum_{i \in V} x_i & \\ \text{s.t.} & \sum_{i \in C} x_i \le 1 & (C \in \mathcal{C}) \\ & x \in \{0,1\}^n. & \end{array} \tag{21}$$

The second competitor is again Gurobi, but using its mixed-integer quadratic convex programming (MIQCP) solver to solve a formulation obtained from the relaxation (20) by declaring all variables to be binary.

It is worth mentioning that Gurobi implements sophisticated strategies to handle convex quadratic inequalities: conversion of quadratic constraints to SOC constraints, linearization of quadratic terms on binary variables, and outer-approximation. The Gurobi parameter `MIQCPMethod` controls the method used to solve MIQCP models. We set it to 1, which 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 cuts). The speed-up factor of Kelley based branch-and-cut w.r.t. the best competitor is also quoted.

We also include Figures 1 and 2, with the performance profiles corresponding to the results of Table 4. The performance profiles, introduced by Dolan and Moré [8], provide a succinct view of the relative behaviour of the different algorithms upon the given test set. Specifically, they plot the probability that the number of evaluated subproblems or the CPU time for an algorithm on a given instance is within a factor of β of the *best* competing algorithm for that instance. Therefore, methods whose corresponding profile lines are the highest are the most effective.

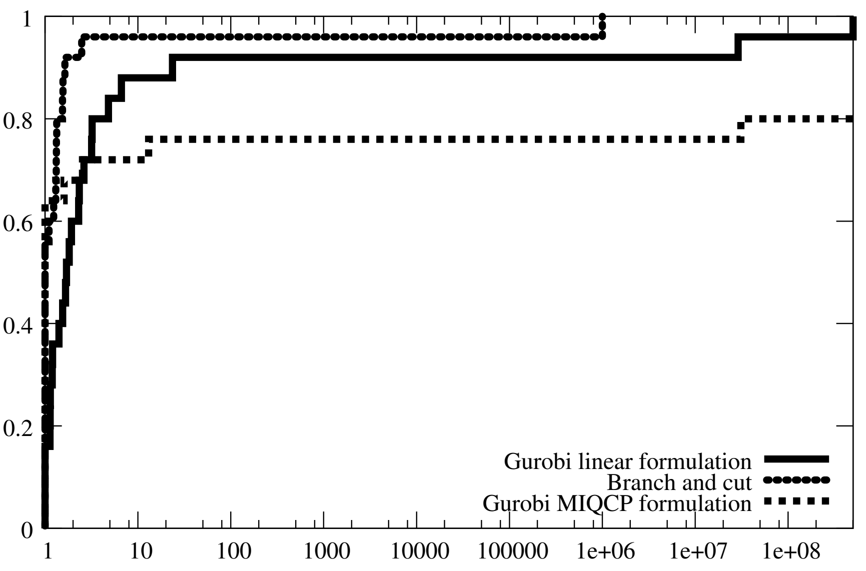

Figure 1: Performance profile of number of subproblems

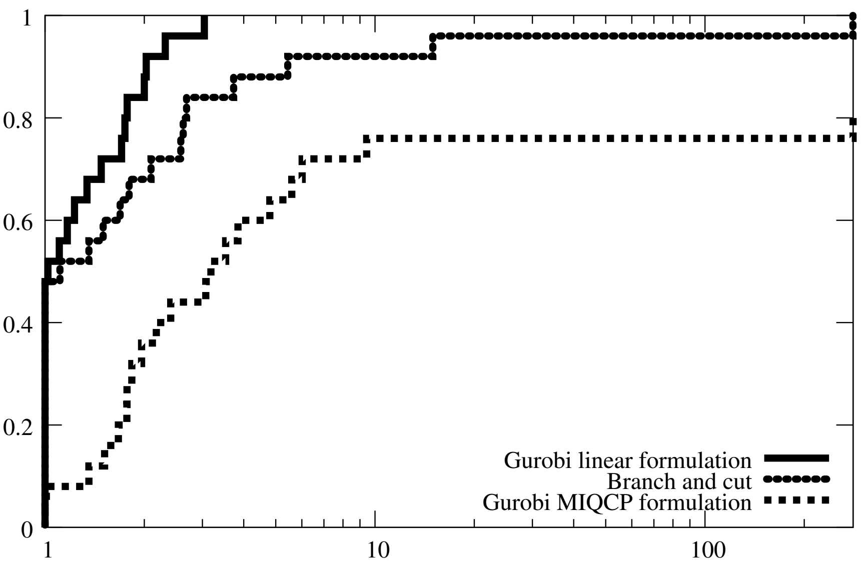

Figure 2: Performance profile CPU time

Figure 1 shows that strengthening the clique formulation, either by the quadratic constraint or by the Kelley cuts generated with Algorithm 1, consistently reduces the size of the enumeration tree. Figure 2 shows a different picture when it comes to CPU time, probably due to the fact that formulation strengthening leads to additional effort in evaluating subproblems. It is still the case that leaving Gurobi to handle internally the"raw" quadratic constraint is generally not cost-effective. In fact, this method fails to solve four large instances within the time limit (`C.250.9`, `brock400` and `p_hat500`, `p_hat700`). On the other hand, branch-and-cut tends to be faster when the Kelley cuts are not included. A more detailed examination, however, shows that Kelley cuts lead to a significant speed-up for some specific families of hard instances, namely, the `brock`, `p_hat` and `sanr` instances.

The overall indication of this experiment is that the proposed Kelley cuts provide a linear reformulation of the quadratic constraint which outperforms 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.

6 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 [36] 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] A. Billionnet, S. Elloumi & M.-C. Plateau (2009) Improving the performance of standard solvers for quadratic 0-1 programs by a tight convex reformulation: The QCR method. *Discr. Appl. Math.*, 157, 1185–1197.

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

[4] 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.

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

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

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

[8] E. Dolan & J. Moré (2002) Benchmarking optimization software with performance profiles. *Math. Program.*, 91, 201–213.

[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.N. 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.) (1996) *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] M. Lobo, L. Vandenberghe, S. Boyd & H. Lebret (1998) Applications of second-order cone programming. *Lin. Alg. App.*, 284, 193–228.

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

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

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

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

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

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

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

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

[33] 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.

[34] 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.

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

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

[37] N. Shor (1987) Quadratic optimization problems. *Sov. J. Comput. Syst. Sci.*, 25, 1–11.

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

	Gurobi clique 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	—
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