Computational Optimization and Applications, 33, 229–247, 2006

DOI: 10.1007/s10589-005-3060-5

On Extracting Maximum Stable Sets in Perfect Graphs Using Lovász's Theta Function*

E. ALPER YILDIRIM† yildirim@bilkent.edu.tr
Department of Industrial Engineering, Bilkent University, 06800 Bilkent, Ankara, Turkey

XIAOFEI FAN-ORZECHOWSKI xfan@ams.sunysb.edu
Department of Applied Mathematics and Statistics, Stony Brook University, Stony Brook, New York

Received June 17, 2004; Revised February 10, 2005; Accepted March 3, 2005

Published online: 18 October 2005

Abstract. We study the maximum stable set problem. For a given graph, we establish several transformations among feasible solutions of different formulations of Lovász's theta function. We propose reductions from feasible solutions corresponding to a graph to those corresponding to its induced subgraphs. We develop an efficient, polynomial-time algorithm to extract a maximum stable set in a perfect graph using the theta function. Our algorithm iteratively transforms an approximate solution of the semidefinite formulation of the theta function into an approximate solution of another formulation, which is then used to identify a vertex that belongs to a maximum stable set. The subgraph induced by that vertex and its neighbors is removed and the same procedure is repeated on successively smaller graphs. We establish that solving the theta problem up to an adaptively chosen, fairly rough accuracy suffices in order for the algorithm to work properly. Furthermore, our algorithm successfully employs a warm-start strategy to recompute the theta function on smaller subgraphs. Computational results demonstrate that our algorithm can efficiently extract maximum stable sets in comparable time it takes to solve the theta problem on the original graph to optimality.

Keywords: maximum stable sets, Lovász's theta function, semidefinite programming, perfect graphs

1. Introduction

Given a simple, undirected graph $G = (V, E)$ with a vertex set $V = \{1, 2, \ldots, n\}$ and an edge set E, where each edge is identified with an unordered pair of its end vertices, a stable set $S \subseteq V$ is a set of mutually nonadjacent vertices. A set $C \subseteq V$ is called a clique if $(i, j) \in E$ for every $i, j \in C$. The maximum stable set (MSS) problem is that of finding a maximum cardinality stable set $S \subseteq V$. We will use $\alpha(G)$ to denote the size of the maximum stable set. A clique cover is the partition of the vertices of G into cliques V_1, $V_2, \ldots, V_k$ such that $V = \cup_{i=1,\ldots,k} V_i$. The problem of finding the smallest number of cliques k, denoted by $\bar{\chi}(G)$, to cover the vertices of G is known as the minimum clique cover (MCC) problem. Since each vertex in a stable set needs to be in a separate clique in any clique cover, it follows that $\alpha(G) \leq \bar{\chi}(G)$.

*This work was supported in part by NSF through CAREER Grant DMI-0237415. Part of this work was performed while the first author was at the Department of Applied Mathematics and Statistics at Stony Brook University, Stony Brook, NY, USA.
†To whom correspondence should be addressed. On leave from the Department of Applied Mathematics and Statistics at Stony Brook University, Stony Brook, NY, USA.

A related problem, known as the maximum clique (MC) problem, is that of finding the largest clique in a given graph. The MC problem on a graph $G = (V, E)$ is equivalent to the MSS problem on the complement of G, denoted by $\overline{G}$, which is a graph obtained from G by removing the existing edges and joining the nonadjacent vertices in G. Another related problem is the graph coloring (GC) problem, which asks to assign the minimum number of colors to vertices of a graph in such a way that no two adjacent vertices receive the same color. The GC problem on a graph $G = (V, E)$ is equivalent to the the MCC problem on $\overline{G}$.

It is well-known that each of the four problems described above is NP-complete in general. For a detailed survey of the MC problem (equivalently the MSS problem), we refer the reader to Bomze et al. [6]. Lovász introduced an invariant of a graph G, known as the Lovász's theta number (henceforth the theta number) and denoted by $\vartheta(G)$, that satisfies the following inequalities [21]:

$$\alpha(G) \leq \vartheta(G) \leq \bar{\chi}(G). \tag{1}$$

$\vartheta(G)$ can be formulated as an optimization problem in several different ways (see Section 2 and also [15] and [20]) and can be computed in polynomial-time via semidefinite programming (SDP).

For a graph $G = (V, E)$ and any $S \subseteq V$, the induced subgraph G_S on S is a graph given by $G_S := (S, E_S)$, where E_S denotes the subset of E that consists only of edges with both end vertices in S. A graph is called *perfect* if $\alpha(G_S) = \bar{\chi}(G_S)$ for all $S \subseteq V$. It follows from (1) that $\alpha(G)$ can be computed in polynomial-time for perfect graphs. Berge conjectured that a graph G is perfect if and only if none of its induced subgraphs is given by an odd cycle of length at least five or its complement [5]. This long standing conjecture, known as the Strong Perfect Graph Conjecture, was recently proved to be true [10]. More recently, a series of papers [9, 11, 12] established that perfect graphs can be recognized in polynomial-time.

For a perfect graph G, in addition to computing $\alpha(G)$, the theta number can also be used to extract a maximum stable set in polynomial time [14]. After computing $\vartheta(G)$, one can delete each node one by one and recompute the theta number in the resulting smaller graph. At each step, by the property of perfect graphs, the theta number either remains the same, in which case the smaller graph still contains a maximum stable set of the same size as the previous graph, or it goes down by one, which implies that the most recently deleted node is in every maximum stable set. Consequently, after at most n computations of the theta number, a maximum stable set can be found in perfect graphs. Currently, this is the only known polynomial-time algorithm for the MSS problem in perfect graphs. The existence of a polynomial-time algorithm of a purely combinatorial nature is still an open problem.

We take a similar approach in this paper in order to develop a practical, polynomial-time algorithm for perfect graphs. The main difference, however, is the exploitation of the properties of an approximate solution of the theta problem, which enables us to effectively identify vertices in a maximum stable set. First, we establish that it suffices to solve the theta problem up to a fairly rough accuracy. Next, we transform the approximate matrix solution of the SDP formulation to an approximate vector solution of another formulation of the theta problem, which can then be used to identify a vertex that belongs to a maximum stable set. Finally, we remove this vertex and all of its neighbors from the graph and continue in an iterative manner starting with the remaining

subgraph. Furthermore, using a reduction among feasible solutions of the theta problem corresponding to a graph and its subgraphs, our algorithm can successfully use a warm-start strategy to recompute the theta number for the subgraphs. The computational results indicate that our algorithm is capable of extracting a maximum stable set in a perfect graph in comparable time it takes to solve the theta problem on the original graph to optimality. The savings in the running time tend to be more significant especially for larger graphs.

Our theoretical contributions include a new transformation of any feasible matrix solution of the SDP formulation of the theta problem into a feasible vector solution of another equivalent formulation with an objective function value no smaller than the original one. This transformation provides a solution to a problem raised in Gruber and Rendl [16]. In addition, we establish that any feasible solution of each of the three formulations of the theta problem presented in Section 2 can be used to obtain a feasible solution of the corresponding formulation of the theta problem on any induced subgraph. This reduction gives rise to an effective warm-start strategy to resolve the theta problem on smaller subgraphs. Finally, we establish several properties of approximate solutions of the theta problem, which play a key role in the design and analysis of our algorithm.

Despite the extensive number of research articles related to Lovász's theta number, [1] very few papers study stable set extractions using the theta number. Grötschel, Lovász, and Schrijver describe several alternative formulations of the theta problem and provide polynomial-time algorithms to compute a (weighted) maximum stable set and a (weighted) minimum clique cover for perfect graphs ([15] Chap. 9) (see also [14]). Alizadeh proposes a Las Vegas type randomized algorithm for the (weighted) MC problem based on perturbing the weight of each vertex [1]. Alon and Kahale present algorithms with performance guarantees to extract large stable sets for graphs with large maximum stable sets as well as for graphs with large theta numbers [2]. Benson and Ye use an alternative SDP formulation to compute the theta number and then apply a random hyperplane strategy with a post-processing stage to extract large stable sets [4]. Burer, Monteiro, and Zhang study the SDP formulation of the theta problem with additional nonconvex low-rank constraints and use continuous optimization techniques to extract large stable sets in fairly large graphs [7]. Gruber and Rendl start by solving the SDP formulation of the theta problem and strengthen it by adding a subset of violated odd-circuit and triangle inequalities based on a transformation of the optimal matrix solution [16]. Consequently, their approach results in a sequence of SDP problems of increasing sizes. In contrast, our approach is based on solving a sequence of SDP problems of successively smaller sizes. On the other hand, their approach can be applied to any imperfect graph G to obtain a sharper bound on $\alpha(G)$ whereas our approach is guaranteed to return a maximum stable set for only perfect graphs.

This paper is organized as follows. In the remainder of this section, we define our notation. Section 2 discusses several formulations of Lovász's theta function and various transformations among feasible solutions of different formulations. We also prove a reduction lemma that forms the basis of the warm-start strategy employed in our algorithm. In Section 3, we study several properties of an approximate solution of the theta problem. In particular, we establish a bound on the required accuracy, which depends on the underlying graph, in order to identify a vertex that belongs to a maximum stable set from an approximate solution of the theta problem. We present our algorithm and

its analysis in Section 4. Section 5 is devoted to computational results and Section 6 concludes the paper.

1.1. Notation

We use $\mathcal{S}^n$ to denote the space of $n \times n$ real symmetric matrices. For two matrices $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times n}$, the trace inner product is denoted by $A \bullet B = \text{trace}(A^T B) = \text{trace}(BA^T) = \sum_{i,j} A_{ij} B_{ij}$. For $A \in \mathcal{S}^n$, we use $A \succeq 0 (A \succ 0)$ to indicate that A is positive semidefinite (positive definite). Note that $d^T A d \geq 0$ ($d^T A d > 0$) for all $d \in \mathbb{R}^n$ ($d \neq 0$) if $A \succeq 0 (A \succ 0)$. Moreover, any principal submatrix of a positive (semi)definite matrix is positive (semi)definite. $A \succeq 0$ if and only if $A = Y^T Y$ for some $Y \in \mathbb{R}^{n\times n}$. The identity matrix is denoted by I, whose dimension will be clear from the context. The vector of all zeroes and the matrix of all zeroes are both represented by 0. We reserve e to denote the vector of all ones in the appropriate dimension and e_j to represent the unit vector whose jth component is 1. We use E_{ij} ($i \neq j$) for the matrix whose (i, j) and (j, i) entries are 1 and remaining entries are 0, i.e., $E_{ij} = e_i {e_j}^T + e_j {e_i}^T$. The matrix of all ones is denoted by J. $\|u\|$ represents the Euclidean norm of $u \in \mathbb{R}^n$. For $U \in \mathbb{R}^{n\times n}$, diag$(U)$ is the vector consisting of the diagonal entries of U. The ceiling of a real number β is denoted by $\lceil \beta \rceil$. The convex hull of a set of vectors $\{d_I, I \in \mathcal{I}\}$ is represented by conv$\{d_I, I \in \mathcal{I}\}$. For a primal-dual pair of optimization problems, we say that an approximate primal-dual solution has absolute error $\epsilon > 0$ if the corresponding duality gap is less than ϵ. In particular, this implies that the objective function values evaluated at such approximate solutions are at most ϵ away from the optimal value. For $S \subseteq \{1, \ldots, n\}$, $\chi^S \in \mathbb{R}^n$ is the incidence vector of S, i.e., $\chi^S_i = 1$ if $i \in S$ and $\chi^S_j = 0$ if $j \notin S$. Given a graph $G = (V, E)$, we denote by $N(i)$ the set of neighbors of $i \in V$, i.e., $N(i) = \{j \in V: (i, j) \in E\}$. The degree of a vertex i is defined as $|N(i)|$. For any $S \subseteq V$, the subgraph of G induced by S is denoted by $G_S = (S, E_S)$.

2. Theta function: Transformations and reductions

In a seminal paper, Lovász proved that a polynomial-time computable invariant of a graph $G = (V, E)$, denoted by $\vartheta(G)$ and known as Lovász's theta number, satisfies (1) [21]. The theta number can be computed through several equivalent formulations [15, 20], three of which are reviewed in this section. The first formulation is an SDP problem:

$$\text{(T1(G))} \quad \vartheta_1(G) := \max_X \{J \bullet X : I \bullet X = 1, X_{ij} = 0, (i, j) \in E, X \succeq 0, X \in \mathcal{S}^n\}.$$

The Lagrangian dual of the SDP problem (T1(G)) is given by

$$\text{(T2(G))} \quad \vartheta_2(G) := \min_{\lambda, y, Z} \left\{ \lambda : -\lambda I + \sum_{(i,j)\in E} y_{ij} E_{ij} + Z = -J, Z \succeq 0, Z \in \mathcal{S}^n \right\}.$$

For a graph $G = (V, E)$, an orthonormal system is a set of unit vectors $\{c, u_i, i \in V\}$ such that ${u_i}^T u_j = 0$ if $(i, j) \notin E$. We define

$$TH(G) := \left\{ x \in \mathbb{R}^n : x \geq 0, \sum_{i \in V} (c^T u_i)^2 x_i \leq 1, \text{ for all orthonormal systems } \{c, u_i, i \in V\} \right\}.$$

An equivalent description of $TH(G)$ is given by a projection onto $\mathbb{R}^n$ of an appropriate subset of the positive semidefinite cone in $\mathcal{S}^{n+1}$ [22]:

$$TH(G) = \left\{ x \in \mathbb{R}^n : \exists W = \begin{bmatrix} U & x \\ x^T & 1 \end{bmatrix} \in \mathcal{S}^{n+1}, \right.$$
$$\left. \operatorname{diag}(U) = x, U_{ij} = 0, (i, j) \in E, W \succeq 0 \right\}.$$

It follows from this definition that $TH(G)$ is a convex set. In fact, $TH(G)$ is a polytope if and only if G is a perfect graph [15], in which case,

$$TH(G) = \operatorname{conv}\{\chi^S \in \mathbb{R}^n : S \subseteq V \text{ is a stable set}\}. \tag{2}$$

Yet another formulation of $\vartheta(G)$ is given by

$$\text{(T3}(G)\text{)} \quad \vartheta_3(G) := \max_x \{e^T x : x \in TH(G)\}.$$

The following result is due to Grötschel et al. [15]:

$$\vartheta_1(G) = \vartheta_2(G) = \vartheta_3(G). \tag{3}$$

While the first equality simply follows from strong duality in SDP (both problems have strictly feasible solutions), the last equality is proved via a transformation of an optimal solution of the SDP problem (T1(G)) to an optimal solution of (T3(G)). We discuss this transformation and several others in the next subsection.

2.1. *Transformations*

Recently, Gruber and Rendl proposed several transformations among feasible solutions of different formulations of the theta function [16]. We start by reviewing some of their transformations in this section. We then propose a new transformation, which provides a solution to a problem left open in Gruber and Rendl [16], and establish its connection with one of their transformations.

First, we discuss their transformation of a feasible solution $\tilde{x}$ of (T3(G)) with $e^T \tilde{x} = \gamma > 0$ to a feasible solution of (T1(G)). Since $\tilde{x} \in TH(G)$, it follows from the description of $TH(G)$ as a projection that there exists a matrix $\widetilde{W} \in \mathcal{S}^{n+1}$ such that

$$\widetilde{W} = \begin{bmatrix} \widetilde{U} & \tilde{x} \\ \tilde{x}^T & 1 \end{bmatrix} \in \mathcal{S}^{n+1}, \quad \operatorname{diag}(\widetilde{U}) = \tilde{x}, \quad \widetilde{U}_{ij} = 0, \quad (i, j) \in E, \quad \widetilde{W} \succeq 0. \tag{4}$$

Therefore, the matrix defined by

$$f(\tilde{x}) := (1/\gamma)\widetilde{U}$$

is a feasible solution of (T1(G)). Furthermore, by (4), $\widetilde{U} - \tilde{x}\tilde{x}^T \succeq 0$, which implies that

$$J \bullet f(\tilde{x}) = (1/\gamma)e^T \widetilde{U} e \geq (1/\gamma)(e^T \tilde{x})^2 = \gamma.$$

When applied to *any* feasible solution of (T3(G)), this transformation yields a feasible solution of (T1(G)) whose objective function value is no smaller. Consequently, if x^* is an optimal solution of (T3(G)), then $X^* := f(x^*)$ is a feasible solution of (T1(G)) such that $J \bullet X^* \geq \vartheta_3(G)$, which implies that $\vartheta_1(G) \geq \vartheta_3(G)$.

In order to establish the reverse inequality towards proving (3), Gruber and Rendl proposed the following elegant transformation of an optimal solution X^* of (T1(G)) to a feasible solution of (T3(G)) [16]:

$$g(X^*) := \vartheta_1(G)\text{diag}(X^*). \tag{7}$$

It follows from (7) that $\vartheta_3(G) \geq e^T g(X^*) = \vartheta_1(G)$, which, together with the previous inequality, establishes (3). Consequently, $g(X^*)$ is in fact an optimal solution of (T3(G)).

Gruber and Rendl note the asymmetry between the transformations (5) and (7). While the former transformation applied to *any* feasible solution of (T3(G)) yields a feasible solution of (T1(G)), the latter transformation can only be applied to an optimal solution X^* of (T1(G)) since it relies on the complementarity property of X^* (see (18) in the proof of Lemma 2.1) that is not necessarily satisfied by arbitrary feasible solutions of (T1(G)). They leave the reverse transformation of an arbitrary feasible solution of (T1(G)) to a feasible solution of (T3(G)) as an open problem. In fact, the straightforward extension of the transformation (7) to any feasible solution X of (T1(G)) given by

$$g(X) := (J \bullet X)\text{diag}(X) \tag{8}$$

may not yield a feasible solution of (T3(G)), as illustrated by the next example.

Example 1. Let $G = (V, E)$ be a 2-path, i.e., $V = \{1, 2, 3\}$ and $E = \{(1, 2), (2, 3)\}$. It is straightforward to verify that

$$X = \begin{bmatrix} 1/3 & 0 & 1/3 \\ 0 & 1/3 & 0 \\ 1/3 & 0 & 1/3 \end{bmatrix}$$

is a feasible solution of (T1(G)). Applying the transformation (8) to X yields $x := g(X) = [5/9, 5/9, 5/9]^T$. We will show that $x \notin TH(G)$. Let c be any unit vector. Let $u_1 = u_2 = c$ and let u_3 be any unit vector orthogonal to c. Then, $\{c, u_i, i = 1, 2, 3\}$ is an orthonormal system for G. However, $\sum_{i=1}^{3} (c^T u_i)^2 x_i = 5/9 + 5/9 = 10/9 \nleq 1$.

In the remainder of this section, we establish the missing link by proposing a reverse transformation, which is a generalization of the transformation used in Grötschel et al. [15]. We also show that this transformation applied to any optimal solution of (T1(G)) reduces to (7).

Proposition 2.1. *Any feasible solution X of (T1(G)) can be transformed into a feasible solution x of (T3(G)) with the property that $e^T x \geq J \bullet X$.*

Proof: Let X be an arbitrary feasible solution of (T1(G)). Since $X \succcurlyeq 0$, there exists a matrix $Y \in \mathbb{R}^{n\times n}$ such that $X = Y^T Y$ (e.g., one can use the Cholesky factorization of X to compute Y or set $Y = X^{1/2}$, the unique symmetric positive semidefinite square root of X). Let y_i denote the ith column of Y. We define

$$\mathcal{P} := \{i \in V : y_i \neq 0\}. \tag{9}$$

Let $v_i := y_i/\|y_i\|$ for $i \in \mathcal{P}$. For $j \notin \mathcal{P}$, choose an orthonormal basis for the orthogonal complement of the subspace spanned by $\{v_i, i \in \mathcal{P}\}$. Let v_j denote the elements of this orthonormal basis. We define

$$d := (1/\sqrt{J \bullet X})Ye. \tag{10}$$

Note that $\|d\| = 1$ since $d^T d = (1/(J \bullet X))\, J \bullet X = 1$. It follows that $\{d, v_i, i \in V\}$ is an orthonormal system for $\overline{G}$ since ${v_i}^T v_j = 0$ if $(i, j) \in E$. For $i \in \mathcal{P}$, we have

$$d^T v_i = \frac{1}{\sqrt{J \bullet X}\|y_i\|} e^T Y^T y_i = \frac{1}{\sqrt{J \bullet X}\|y_i\|} e^T X e_i \tag{11}$$

so that $\|y_i\|\ (d^T v_i) = (1/\sqrt{J \bullet X}) e^T X e_i$ for $i \in \mathcal{P}$. Summing over all $i \in V$, we obtain

$$\sum_{i\in V} \|y_i\|(d^T v_i) = (1/\sqrt{J \bullet X}) e^T X e = \sqrt{J \bullet X}. \tag{12}$$

Finally, an application of the Cauchy-Schwarz inequality yields

$$\begin{aligned} J \bullet X &= \left(\sum_{i\in V} \|y_i\|(d^T v_i)\right)^2 \leq \left(\sum_{i\in V} \|y_i\|^2\right)\left(\sum_{i\in V}(d^T v_i)^2\right) \\ &= \text{trace}(X)\left(\sum_{i\in V}(d^T v_i)^2\right) = \sum_{i\in V}(d^T v_i)^2. \end{aligned} \tag{13}$$

We now define a transformation $h(X) = x$ by

$$x_i := (d^T v_i)^2, \quad i \in V. \tag{14}$$

We will show that $x \in TH(G)$. Let $\{c, u_i, i \in V\}$ be any orthonormal system for G. It follows that

$$(u_i v_i)^T \bullet (u_j v_j)^T = \left(u_i^T u_j\right)\left(v_i^T v_j\right) = \begin{cases} 1, & \text{if } i = j, \\ 0, & \text{otherwise,} \end{cases} \tag{15}$$

which implies that the matrices $\{u_i\ v_i{}^T,\ i \in V\}$ are mutually orthogonal and have unit norm with respect to the trace inner product. Hence,

$$1 = (cd^T) \bullet (cd^T) \geq \sum_{i \in V} \left((cd^T) \bullet \left(u_i v_i^T\right)\right)^2 = \sum_{i \in V} (c^T u_i)^2 x_i, \tag{16}$$

which shows that $x \in TH(G)$. It follows from (13) that $e^T\ x \geq J \bullet X$. □

We remark that the auxiliary vectors $\{d, v_i, i \in V\}$ need not be computed explicitly for the aforementioned transformation. In fact, one has (cf. (14))

$$x_i = (d^T v_i)^2 = \frac{1}{(J \bullet X) X_{ii}} \left(\sum_{j=1}^{n} X_{ij} \right)^2 \tag{17}$$

if $i \in \mathcal{P}$ and $x_i = 0$ otherwise. We use this observation in our implementation. This transformation applied to the feasible solution X of Example 1 yields $x := h(X) = [4/5, 1/5, 4/5]^T = (4/5)\ [1, 0\ ,1]^T + (1/5)\ [0, 1, 0]^T \in TH(G)$ by (2) and $e^T\ x = 9/5 \geq 5/3 = J \bullet X$.

We conclude this section by showing that the transformation (14) applied to any optimal solution of (T1(G)) reduces to (7).

Lemma 2.1. *Let X^* be an optimal solution of the SDP problem (T1(G)). Let $g(X^*)$ and $h(X^*)$ denote the two transformations of X^* given by (7) and (14), respectively. Then, $g(X^*) = h(X^*)$.*

Proof: Let $\tilde{x} := g(X^*)$ and $x^* := h(X^*)$. Our first step is to establish that $\tilde{x} = X^* e = \vartheta_1(G)\text{diag}(X^*)$. We follow the argument in Gruber and Rendl [16]. Let (λ^*, y^*, Z^*) be an optimal solution for the SDP problem (T2(G)). By strong duality, we have $X^*\ Z^* = 0$ and $\lambda^* = \vartheta_1(G)$. Note that

$$Z^*_{ij} = \begin{cases} \vartheta_1(G) - 1, & \text{if } i = j, \\ -1, & \text{if } (i, j) \notin E, \\ -1 - y^*_{ij}, & \text{if } (i, j) \in E. \end{cases}$$

Thus,

$$\begin{aligned} (X^* Z^*)_{ii} &= X^*_{ii} Z^*_{ii} + \sum_{(i,j) \in E} X^*_{ij} Z^*_{ij} + \sum_{(i,j) \notin E, i \neq j} X^*_{ij} Z^*_{ij}, \\ &= (\vartheta_1(G) - 1) X^*_{ii} - \sum_{(i,j) \notin E, i \neq j} X^*_{ij}, \end{aligned} \tag{18}$$

which implies that $\tilde{x}_i = (X^* e)_i = \vartheta_1(G)\ X^*_{ii}$, or equivalently that $\tilde{x} = X^* e = \vartheta_1(G)\text{diag}(X^*)$.

Let $X^* = Y^T Y$, where $Y = [y_1, \ldots, y_n]$. By definition of $\{d, v_i, i \in V\}$ in the proof of Proposition 2.1, $x_j^* = 0$ if $j \notin \mathcal{P}$, which implies that $y_j = 0$, or equivalently, $X_{jj}^* = 0$. Hence, $\tilde{x}_j = 0$. Otherwise, by (11),

$$x_i^* = (d^T v_i)^2 = \frac{1}{\vartheta_1(G) X_{ii}^*} \vartheta_1(G)^2 (X_{ii}^*)^2 = \vartheta_1(G) X_{ii}^*,$$

where we used $X^* e = \vartheta_1(G) \mathrm{diag}(X^*)$. This completes the proof. □

2.2. *Reductions*

We now show that any feasible solution of the optimization problems (T1(G))—(T3(G)) can be reduced to a feasible solution of the corresponding problem for the induced subgraph G_S for any $S \subseteq V$. The next lemma plays a crucial role in developing a warm-start strategy in our implementation.

Lemma 2.2. *Let $G = (V, E)$ be a graph and let $S \subseteq V$, $S \neq \emptyset$. Then any feasible solution of the optimization problems (T1(G)), (T2(G)), and (T3(G)) can be transformed into a feasible solution of the optimization problems (T1(G_S)), (T2(G_S)), and (T3(G_S)), respectively.*

Proof: Let X be a feasible solution of (T1(G)). Let $M := X(S, S)$ denote the submatrix of X whose rows and columns are indexed by the indices in S. Then, $M_{kl} = 0$ if $(k, l) \in E_S$ and $M \succcurlyeq 0$ since $X \succcurlyeq 0$. If $\mathrm{trace}(M) = 0$, then $M = 0$. In this case, we can use the obvious feasible solution $\widetilde{M} = (1/|S|)I$. Otherwise, we only need to scale M to satisfy the trace constraint, i.e., $\widetilde{M} = (1/\mathrm{trace}(M))\, M$ is a feasible solution of (T1(G_S)).

Let (λ, y, Z) be a feasible solution of (T2(G)). We claim that $(\lambda, y_{E_S}, Z(S, S))$ is a feasible solution of (T2(G_S)), where y_{E_S} is the restriction of y to indices $(k, l) \in E_S$. This is easily verified by restricting the matrix equality constraint in (T2(G)) to the submatrix indexed by S. By a similar argument, $Z(S, S) \succcurlyeq 0$.

Finally, let $\tilde{x} \in TH(G)$. We claim that $\tilde{x}_S \in TH(G_S)$, where $\tilde{x}_S$ is the restriction of $\tilde{x}$ to indices in S. Since $\tilde{x} \in TH(G)$, there exists a matrix $\widetilde{W}$ that satisfies the conditions in (4). It is easily argued that the reduced matrix

$$\widetilde{W}_S := \begin{pmatrix} \widetilde{U}(S, S) & \tilde{x}_S \\ \tilde{x}_S^T & 1 \end{pmatrix} \succeq 0$$

satisfies the requirements to yield a feasible solution $\tilde{x}_S$ of (T3(G_S)), which implies that $\tilde{x}_S \in TH(G_S)$. □

We remark that the reductions of Lemma 2.2 preserve optimality under certain conditions. If X^* and (y^*, Z^*) are optimal solutions of (T1(G)) and (T2(G)), respectively, and if $X^*_{il} = 0$ for $i \notin S$, then the reduced solutions remain optimal for (T1(G_S)) and (T2(G_S)), respectively. Similarly, if x^* is an optimal solution of (T3(G)) and $x^*_i = 0$ for $i \notin S$, then the reduced solution remains optimal for (T3(G_S)).

3. Properties of approximate solutions

In this section, we derive several properties of a near-optimal feasible solution of (T3(G)) for a perfect graph G. The results of this section will be used in the algorithm to identify a vertex that belongs to a maximum stable set from an approximate solution of (T3(G)).

Given a perfect graph $G = (V, E)$, let

$$\mathcal{S} := \{S \subseteq V : S \text{ is a stable set}, |S| = \alpha(G)\}, \tag{19}$$

i.e., $\mathcal{S}$ is the collection of all maximum stable sets of G. Similarly, let

$$\mathcal{T} := \{T \subseteq V : T \text{ is a stable set}, |T| < \alpha(G)\}, \tag{20}$$

i.e., $\mathcal{T}$ is the collection of all other stable sets of G, including the empty set.

Next, we define the following index sets.

$$I := \{i \in V : \exists\, S \in \mathcal{S} \text{ such that } i \in S\}, \tag{21}$$

$$J := \{j \in V : j \notin S \text{ for all } S \in \mathcal{S}\}. \tag{22}$$

The index sets I and J partition the vertex set V based on whether each vertex belongs to at least one maximum stable set of G.

Our first result provides an upper bound on the components of a near-optimal feasible solution $x \in TH(G)$ corresponding to indices in J.

Proposition 3.1. *Let $G = (V, E)$ be a perfect graph and $\epsilon \in [0, 1]$. Suppose that $x \in TH(G)$ satisfies $e^T x \geq \alpha(G) - \epsilon$. Then, $x_j \leq \epsilon$ for all $j \in J$, where J is defined by (22).*

Proof: We use the characterization of $TH(G)$ given by (2). We can therefore write $x \in TH(G)$ as a convex combination of the incidence vectors of stable sets of G, i.e.,

$$x = \sum_{S \in \mathcal{S}} \lambda_S \chi^S + \sum_{T \in \mathcal{T}} \lambda_T \chi^T, \tag{23}$$

where $\lambda_S \geq 0$ for all $S \in \mathcal{S}$, $\lambda_T \geq 0$ for all $T \in \mathcal{T}$, and

$$\sum_{S \in \mathcal{S}} \lambda_S + \sum_{T \in \mathcal{T}} \lambda_T = 1. \tag{24}$$

Multiplying both sides of (23) by e^T, we obtain

$$e^T x = \sum_{S\in\mathcal{S}} \lambda_S e^T \chi^S + \sum_{T\in\mathcal{T}} \lambda_T e^T \chi^T \le \alpha(G) \sum_{S\in\mathcal{S}} \lambda_S + (\alpha(G)-1) \sum_{T\in\mathcal{T}} \lambda_T,$$
$$= \alpha(G) \left(\sum_{S\in\mathcal{S}} \lambda_S + \sum_{T\in\mathcal{T}} \lambda_T \right) - \sum_{T\in\mathcal{T}} \lambda_T = \alpha(G) - \sum_{T\in\mathcal{T}} \lambda_T,$$

where we used $e^T\chi^T = |T| \le \alpha(G) - 1$ for all $T \in \mathcal{T}$ in the first inequality and (24) in the last equality.

By the hypothesis, we have $e^T x \ge \alpha(G) - \epsilon$, which, together with the previous inequality, implies that

$$\sum_{T\in\mathcal{T}} \lambda_T \le \epsilon. \tag{25}$$

However, since no vertex $j \in J$ belongs to a maximum stable set, we obtain

$$x_j = \sum_{T\in\mathcal{T}} \lambda_T \chi^T_j \le \sum_{T\in\mathcal{T}} \lambda_T, \quad \forall j \in J.$$

Combining this inequality with (25) completes the proof. □

Proposition 3.1 implies that the components of a near-optimal $x \in TH(G)$ corresponding to indices in J are small. It follows from (2) that the set of optimal solutions of (T3(G)) is simply the convex hull of the incidence vectors χ^S of $S \in \mathcal{S}$. Therefore, Proposition 3.1 serves also as a verification of the obvious result that $x^*_j = 0$ for all $j \in J$ at any optimal solution x^* of (T3(G)).

Furthermore, Proposition 3.1 leads to the following immediate result.

Corollary 3.1. *Let $G = (V, E)$ be a perfect graph and $\epsilon \in [0, 1)$. Suppose that $x \in TH(G)$ satisfies $e^T x \ge \alpha(G) - \epsilon$. Then, the set*

$$K := \{i \in V : x_i > \epsilon\} \tag{26}$$

satisfies $K \subseteq I$, where I is defined by (21).

We remark that Corollary 3.1 provides only a partial characterization of the index set I. In particular, there may be vertices $i \in I$ such that $x_i \le \epsilon$. In the extreme case, K may be empty if ϵ is sufficiently large. Therefore, in order to identify at least one vertex in I via the set K, we need to select ϵ carefully. The following result gives an upper bound on ϵ so as to ensure that K is nonempty.

Proposition 3.2. *Let $G = (V, E)$ be a perfect graph. Suppose that $x \in TH(G)$ satisfies $e^T x \ge \alpha(G) - \epsilon$. If $\epsilon < \alpha(G)/(n+1)$, then the set K defined by (26) is nonempty.*

Proof: By Proposition 3.1, $x_j \leq \epsilon$ for all $j \in J$. Therefore, $\sum_{j\in J} x_j \leq |J|\,\epsilon$. It follows that

$$\begin{aligned}\sum_{i\in I} x_i &= e^T x - \sum_{j\in J} x_j, \\ &\geq \alpha(G) - \epsilon(|J| + 1).\end{aligned}$$

Since the maximum component of x in I is at least as large as the average of the corresponding components, we obtain

$$\max_{i\in I} x_i \geq \frac{\alpha(G) - \epsilon(|J| + 1)}{|I|}.$$

A sufficient condition to ensure that K is nonempty is given by

$$\frac{\alpha(G) - \epsilon(|J| + 1)}{|I|} > \epsilon.$$

Solving the inequality for ϵ together with $|I| + |J| = n$ proves the assertion. □

Proposition 3.2 provides a sufficient condition to ensure that a vertex $i \in I$ can be identified from a near-optimal solution $x \in TH(G)$. Our algorithm relies on this result to select the parameter ϵ in an adaptive manner. We conclude this section by pointing out that neither of the bounds in Propositions 3.1 and 3.2, in general, can be improved.

Example 2. Let $G = (V, E)$ denote the graph in Example 1. For $\epsilon \in [0,1]$, consider the following family of feasible solutions of $TH(G)$ parametrized by ϵ:

$$x(\epsilon) := \epsilon[0, 1, 0]^T + (1 - \epsilon)[1, 0, 1]^T.$$

Note that $\alpha(G) = 2$ and $e^T x(\epsilon) = 2 - \epsilon$. Since $I = \{1, 3\}$ and $J = \{2\}$, we have $x_2 = \epsilon \leq \epsilon$. Furthermore, the set K defined by (26) is nonempty and coincides with I if and only if $\epsilon < 1 - \epsilon$, or $\epsilon < 1/2 = \alpha(G)/(n + 1)$.

4. The algorithm

In this section, we present an algorithm to extract a maximum stable set from a perfect graph using Lovász's theta function. Our algorithm is driven by the results of Section 2 and Section 3.

We now briefly explain Algorithm 1 outlined in the next page. The input is a perfect graph $G = (V, E)$. After initializing S (line 1), we preprocess G to put all the isolated and degree-1 vertices of G into S and remove them as well as their neighbors from G (lines 2–6). Clearly, any isolated vertex belongs to all maximum stable sets of G. A simple swapping argument shows that a degree-1 vertex can also be included in a maximum stable set. The preprocessing step may reduce the size of the graph. At step 8, we solve

Algorithm 1 Maximum Stable Set Algorithm for Perfect Graphs
Require: $G = (V, E)$, a perfect graph.

1: $S \leftarrow \emptyset$.
2: **while** G has isolated and/or degree-1 vertices **do**
3: **loop**
4: if there exists an isolated vertex $i \in V$, then $S \leftarrow S \cup \{i\}$, $V \leftarrow V \backslash \{i\}$, $G \leftarrow (V, E)$.
5: if there exists a degree-1 vertex $j \in V$, then $S \leftarrow S \cup \{j\}$, $V \leftarrow V \backslash (\{j\} \cup N(j))$, $E \leftarrow E_V$, $G \leftarrow (V, E)$.
6: **end loop**
7: $G \leftarrow (V, E)$, $\epsilon \leftarrow .99$.
8: Solve (T1(G)) and (T2(G)) to absolute error ϵ and store the approximate solution (X, y, Z).
9: $\alpha \leftarrow \lceil C \bullet X \rceil$.
10: **while** $|V| > 0$ **do**
11: **loop**
12: $n \leftarrow |V|$, $\epsilon \leftarrow \alpha/(n+1)$. Solve (T1($G$)) and (T2($G$)) starting from (X, y, Z) to absolute error ϵ and store the approximate solution $(\widetilde{X}, \tilde{y}, \widetilde{Z})$.
13: Transform $\widetilde{X}$ to a feasible solution $\tilde{x}$ of (T3 (G)) using (17).
14: $i \leftarrow \arg\max_{k \in K}\{|N(k)|\}$, where K is defined by (26).
15: $S \leftarrow S \cup \{i\}$, $V \leftarrow V \backslash (\{i\} \cup N(i))$, $E \leftarrow E_V$, $\alpha \leftarrow \alpha - 1$, $G \leftarrow (V, E)$.
16: **while** G has isolated and/or degree-1 vertices **do**
17: **loop**
18: if there exists an isolated vertex $i \in V$, then $S \leftarrow S \cup \{i\}$, $V \leftarrow V \backslash \{i\}$, $\alpha \leftarrow \alpha - 1$, $G \leftarrow (V, E)$.
19: if there exists a degree-1 vertex $j \in V$, then $S \leftarrow S \cup \{j\}$, $V \leftarrow V \backslash (\{j\} \cup N(j))$, $E \leftarrow E_V$, $\alpha \leftarrow \alpha - 1$, $G \leftarrow (V, E)$.
20: **end loop**
21: Set (X, y, Z) to the reduced solution obtained from $(\widetilde{X}, \tilde{y}, \widetilde{Z})$ via Lemma 2.2.
22: **end loop**
23: Output S, $|S|$.

the theta problem to an absolute error less than one in order to determine $\alpha(G)$ (line 9) and store the corresponding primal and dual solutions. Then, the main loop is executed. The parameters n and ϵ are set adaptively based on the current graph (line 12). We resolve the theta problem up to an absolute error ϵ using the previously stored solution as a warm-start. The resulting approximate primal solution is then transformed into a solution of (T3(G)), which is used to identify a vertex to be added to S. Among such vertices, the algorithm picks one with the largest number of neighbors in an attempt to minimize the number of vertices in the remaining subgraph. After removing that vertex and all of its neighbors from G, the preprocessing step is rerun to possibly eliminate further isolated and degree-1 vertices. The parameters ϵ and α are updated accordingly and the theta function is recomputed on the smaller subgraph using the reduced solution as a warm-start. The next theorem provides an analysis of the complexity of Algorithm 1.

Theorem 4.1 *Let $G = (V, E)$ be a perfect graph. Algorithm 1 terminates in polynomial-time with a maximum stable set $S \subseteq V$ after at most min $\{\alpha(G), n/3\}$ approximate theta number computations on graphs of successively smaller sizes.*

Proof: Clearly, after line 9, $\alpha(G)$ is already computed. During the execution of the main loop, ϵ is set in such a way that the transformation of an approximate solution of (T1(G)) via (17) can be used to identify a vertex that belongs to a maximum stable set by Propositions 2.1 and 3.2. The preprocessing step eliminates the remaining isolated and degree-1 vertices. Therefore, at each theta number computation, the minimum degree of any vertex in the underlying graph is at least two, which implies that at least three vertices are removed from G every time line 15 is executed. Since every induced subgraph of a perfect graph is also perfect and the theta number can be computed to within arbitrary absolute error in polynomial-time using interior-point methods, the assertion follows□

Since the MSS problem is in general NP-complete for imperfect graphs, Algorithm 1 does not necessarily return a maximum stable set upon termination for such graphs. In particular, the results of Section 3 do not apply to the case of an imperfect graph $G = (V, E)$ since the relation (2) fails to hold. While the inequality $\alpha(G) \leq \vartheta(G)$ is still satisfied, $\vartheta(G)$ can be a fairly poor upper bound for $\alpha(G)$. In fact, for every $\epsilon > 0$, Feige constructed families of imperfect graphs on n vertices such that $\vartheta(G) > n^{1-\epsilon}\alpha(G)$ [13]. This is not a surprising result since Hastad proved that $\alpha(G)$ cannot be approximated to within a factor of $O(n^{1-\epsilon})$ unless any problem in NP admits a probabilistic polynomial-time algorithm [17]. This observation in conjunction with the inherent limitations of the existing algorithms for semidefinite programming suggests that an algorithm similar to Algorithm 1 for general graphs is unlikely to be competitive with the several existing efficient heuristic approaches. Therefore, we exclusively restrict our analysis to perfect graphs in this paper.

5. Computational results

In this section, we discuss some details of the implementation of the algorithm outlined in the previous section and present the results on some well-documented graphs.

Since our algorithm is specifically designed for perfect graphs, a first prerequisite is to locate instances of such graphs in order to test our implementation. Despite the fact that perfect graphs can be recognized in polynomial time [9, 12, 13], the running time of the algorithm is $O(n^9)$ and is not practical even for small graphs. Therefore, generating a random graph and testing for perfectness is not a viable option. Another option is to generate random graph instances from certain known classes of perfect graphs (e.g., line graphs of bipartite graphs, interval graphs). However, given that there are at least 96 known classes of perfect graphs [19], such an experiment would be fairly restrictive in nature.

Instead of using randomly generated graphs, we decided to test Algorithm 1 on well-documented instances. However, we were unable to locate an exclusive collection of instances of perfect graphs. Therefore, we decided to use the clique and coloring instances from the Second DIMACS Implementation Challenge[2] and from the instances of graph coloring problems collected by Trick.[3] We also included a few line graphs tested in Benson and Ye [4] for the MSS problem. For the DIMACS problems,

maximum stable sets and theta numbers are known for all but a few very large instances (see, e.g., the DIMACS web site or Burer et al. [7]). For the remaining graphs, we used the following criterion. We included an instance in our test set only if it has an integral theta number. We stress that this is only a necessary condition in order for a graph to be perfect. In fact, some of the graphs included in our test set such as the Mycielski graphs [23] are known to be imperfect (and yet have $\alpha(G) = \vartheta(G)$). This is a difficult class of graphs since the size of a maximum clique is two whereas the chromatic number increases with their size. Nevertheless, Algorithm 1 successfully computed a stable set on all instances whose size matches with the corresponding theta number. It follows from (1) that the computed stable sets are indeed maximal.

We implemented Algorithm 1 in MATLAB 6.5 in exactly the same way as it is presented in Section 4. Each semidefinite programming (SDP) problem was solved using SDPT3 version 3.0 [25], a primal-dual path-following interior-point solver for semidefinite programming. We used the computationally more efficient formulation (T1(G)) in our experiments since the SDP formulation of (T3(G)) requires $n + 1$ additional equality constraints. SDPT3 includes a subroutine called thetaproblem that sets up the theta problem and computes the theta number via (T1(G)) using the adjacency matrix of a graph as input. We used this subroutine to set up the theta problems. We slightly modified the termination criterion in the solver in order to account for a prespecified absolute error as opposed to the default relative error. After the theta problem was properly set up, we called the main solver sqlp, which enables the user to take advantage of warm-starts by specifying a starting point. All of the computations were performed on a 1.7 GHz Pentium IV processor with 512 MB of RAM running under Linux.

Table 1 presents the results of the implementation on forty-five instances. The rows are divided into four major groups based on the origin of the instances. The first group consists of the complements of the instances for the maximum clique problem from the Second DIMACS Implementation Challenge. The line graphs used in Benson and Ye [4] constitute the second group. The third group contains the Mycielski graphs [23]. The coloring instances from Trick's website constitute the last group. Table 1 consists of seven columns divided into four groups. The first column presents the name of the instance. The second group of columns reports the number of nodes $|V|$, the number of edges $|E|$, and the size of the maximum stable set $\alpha(G)$, which coincides with $\vartheta(G)$. The third group of columns presents the CPU times in seconds rounded to the nearest integer. (T_1) denotes the running time of the theta function using the subroutine thetaproblem in SDPT3 with the default tolerances (i.e., 10^{-8} both for the relative duality gap and feasibility). (T_2) represents the running time of Algorithm 1 on the corresponding instance. Finally, the last column presents the number of theta function computations (i.e., number of times the while loop is executed) by Algorithm 1.

Table 1 indicates that Algorithm 1 is capable of extracting maximum stable sets in graphs of size up to several hundred vertices and several thousand edges in a fairly short amount of time. In particular, Algorithm 1 terminated in less than about 20 minutes on all of the tested instances and in less than 5 minutes on most of the instances. A comparison of the columns T_1 and T_2 reveals that the running time of Algorithm 1 is comparable to the computation time of the theta number on the original graph. In fact, Algorithm 1 was slower only on sixteen of the forty-five graphs and outperformed the theta computation on most of the larger graphs. Line graphs were the only instances for which it was significantly slower. On this class of graphs, Stephen and Tunçel established

Table 1. Computational results.

	Graph			Time		
Instance	$\|V\|$	$\|E\|$	$\alpha(G)$	T_1	T_2	Num.
hamming6-2.co	64	192	32	2	4	5
hamming8-2.co	256	1024	128	14	40	16
johnson8-2-4.co	28	168	4	1	1	3
johnson8-4-4.co	70	560	14	4	7	6
johnson16-2-4.co	120	1680	8	24	27	7
san200-0.7-1.co	200	5970	30	1402	1208	4
san200-0.9-1.co	200	1990	70	53	53	6
san200-0.9-2.co	200	1990	60	61	62	7
san200-0.9-3.co	200	1990	44	89	71	8
line1	248	1202	50	23	84	25
line4	597	3486	100	281	1202	58
line5	597	3481	100	297	1146	57
line6	597	3625	100	325	1246	55
myciel3	11	20	5	2	1	1
myciel4	23	71	11	1	1	2
myciel5	47	236	23	2	3	3
myciel6	95	755	47	7	9	4
myciel7	191	2360	95	84	69	5
queen5.5	25	160	5	2	2	2
queen6.6	36	290	6	2	3	3
queen7.7	49	476	7	4	4	4
queen8.8	64	728	8	5	7	4
queen9.9	81	1056	9	10	11	5
queen10.10	100	1470	10	21	19	5
queen11.11	121	1980	11	42	40	6
queen12.12	144	2596	12	82	73	7
queen13.13	169	3328	13	159	138	8
queen14.14	196	4186	14	292	247	8
anna	138	493	80	8	2	5
david	87	406	36	5	6	8
huck	74	301	27	4	4	7
jean	80	254	38	4	2	4
games120	120	638	22	7	18	12
miles250	128	387	44	6	11	13
miles750	128	2113	12	63	54	6
miles1000	128	3216	8	180	123	4

Table 1. Continued.

	Graph			Time		
Instance	$\|V\|$	$\|E\|$	$\alpha(G)$	T_1	T_2	Num.
miles1500	128	5198	5	690	589	3
zeroin.i.1	211	4100	120	334	232	4
zeroin.i.2	211	3541	127	239	162	7
zeroin.i.3	206	3540	123	239	162	7
mulsol.i.1	197	3925	100	288	177	4
mulsol.i.2	188	3885	90	280	213	7
mulsol.i.3	184	3916	86	284	211	6
mulsol.i.4	185	3946	86	291	222	7
mulsol.i.5	186	3973	88	296	224	6

that successive relaxation methods based on SDP perform poorly for the MSS problem in the worst case [24].

Several factors contribute to the efficiency of Algorithm 1. First of all, a close examination of the last column in Table 1 reveals that the number of calls to the theta function by Algorithm 1 is usually much smaller than the worst-case theoretical bound of Theorem 4.1. This is mainly due to the fact that Algorithm 1 removes a vertex with the largest degree, which results in a potentially significant reduction in the size of the subsequent graphs. Line graphs are the only instances that require relatively large number of theta function computations. Secondly, Algorithm 1 computes only an approximate solution. For our test set, the computed solution usually is correct up to only three digits of accuracy since $n < 1000$ on all of the tested instances (cf. Proposition 3.2), which results in substantial savings in terms of the number of interior-point iterations in comparison with the default value of eight digits of accuracy. Thirdly, Algorithm 1 significantly reduces the computation time of the theta number for smaller subgraphs by utilizing a warm-start strategy. Interior-point algorithms are known to be very sensitive to the initial iterate. Since each theta function is solved to a fairly rough accuracy, the reduced solution tends to be sufficiently far from the boundary of the feasible region, thereby providing a very good starting point for the subsequent theta number computation. In practice, we observed that it typically takes only two to three interior-point iterations to resolve the theta problem up to the specified accuracy. Finally, the preprocessing stage helps to yield further reduction in the size of the problem.

We do not compare Algorithm 1 with the other available efficient heuristic approaches for the following reasons. Our main focus in this paper is to develop a practical algorithm for perfect graphs without sacrificing polynomiality. The only polynomial-time algorithm we are aware of is due to Grötschel et al. [14], which is based on the computationally-not-so-efficient ellipsoid algorithm. On the other hand, there are several efficient heuristics which can handle much larger instances than our algorithm (see, e.g., the references in Bomze et al. [6] and Burer et al. [7]). The only bottleneck of Algorithm 1 is the computation of the theta number using the SDP formulation with a $|V| \times |V|$ matrix and $|E| + 1$ equality constraints (cf. T1(G)). The current solvers can

handle graphs of up to a few thousand vertices and a few thousand edges. In fact, we were able to solve the theta problem on several larger instances on the same personal computer from the Second DIMACS Challenge such as MANN-a45.co (1035, 1980), brock200-4.co (200, 5066), keller4.co (171, 5100), where $(\cdot,\cdot)$ denotes the number of vertices and edges, respectively, However, all of these instances failed to produce an integral theta number, which is a certificate of imperfectness. Therefore, we did not include these instances in our results. Nevertheless, we stress that Algorithm 1 can compute in polynomial-time a maximum stable set in a perfect graph whereas the heuristic approaches do not have any theoretical guarantees. As a result, we do not think that a comparison would be meaningful. Rather, we view Algorithm 1 as a complement to the existing rich literature on the maximum stable set problem.

6. Concluding remarks

In this paper, we presented a practical polynomial-time algorithm to extract a maximum stable set in perfect graphs using Lovász's theta function. Our algorithm relies on various transformations and reductions among different formulations of the theta problem and on several new properties of the near-optimal feasible solutions of the appropriate formulation of the theta problem. The algorithm sequentially computes the theta function up to an adaptively selected rough accuracy on successively smaller graphs and effectively employs a warm-start strategy. Computational results indicate that our algorithm can efficiently extract maximum stable sets on several test instances.

One disadvantage of our algorithm is the extensive memory requirements of interior-point methods for SDP on large-scale graphs. Bundle methods [18] or recent nonlinear optimization algorithms for SDP [3, 8] may lead to the successful application of our algorithms to larger perfect graphs—especially since only a rough approximation is required at each computation of the theta number. We intend to explore similar alternatives.

The minimum clique cover (equivalently graph coloring) problem can also be solved in polynomial-time for perfect graphs [14]. In the near future, we intend to work on a similar, practical, polynomial-time algorithm for the minimum clique cover problem relying on an approximate solution of the theta function.

Acknowledgments

We are grateful to Steven Benson for sharing the line graphs used in their experiments and for his help with a modification of DSDP3 in the early stages of this research. We also thank Franz Rendl for his insightful comments and suggestions on an earlier draft of this manuscript. We gratefully acknowledge insightful comments from two anonymous referees.

Notes

1. The Science Citation Index lists about 150 papers citing Lovász's original paper as of June 2004.
2. http://mat.gsia.cmu.edu/challenge.html
3. http://mat.gsia.cmu.edu/COLOR/instances.html

References

1. F. Alizadeh, "A sublinear-time randomized parallel algorithm for the maximum clique problem in perfect graphs," ACM-SIAM Symposium on Discrete Algorithms, vol. 2, pp. 188–194, 1991.
2. N. Alon and N. Kahale, "Approximating the independence number via the theta-function," Mathematical Programming, vol. 80, no. 3. pp. 253–264, 1998.
3. H.Y. Benson and R.J. Vanderbei, "Solving problems with semidefinite and related constraints using interior-point methods for nonlinear programming," Mathematical Programming, vol. 95, no. 2, pp. 279–302, 2003.
4. S. Benson and Y. Ye, "Approximating maximum stable set and minimum graph coloring problems with the positive semidefinite relaxation," in Applications and Algorithms of Complementarity, M. Ferris and J. Pang (Eds.), Kluwer Academic Publishers, 2000, pp. 1–18.
5. C. Berge, "Färbung von graphen deren sämtliche beziehungsweise deren ungerade kreise starr sind (zusammenfassung)," Wissenschaftliche Zeitschrift, Martin Luther Univ. Halle-Wittenberg, Math.-Naturwiss. Reihe, pp. 114–115, 1961.
6. I.M. Bomze, M. Budinich, P.M. Pardalos, and M. Pelillo, "The maximum clique problem," in Handbook of Combinatorial Optimization (Supplement Volume A), D.-Z. Du and P.M. Pardalos (Eds.), Kluwer Academic, Boston, Massachusetts, U.S.A., 1999, pp. 1–74.
7. S. Burer, R.D.C. Monteiro, and Y. Zhang, "Maximum stable set formulations and heuristics based on continuous optimization," Mathematical Programming, vol. 94, no.1, pp. 137–166, 2002.
8. S. Burer, R.D.C. Monteiro, and Y. Zhang, "Solving a class of semidefinite programs via nonlinear programming," Mathematical Programming, vol. 93, no. 1, pp. 97–122, 2002.
9. M. Chudnovsky, G. Cornuejols, X. Liu, P. Seymour, and K. Vuskovic, "Cleaning for Bergeness," Technical report, Princeton University, 2003.
10. M. Chudnovsky, N. Robertson, P. Seymour, and R. Thomas, "The strong perfect graph theorem," Technical report, Princeton University, 2003.
11. M. Chudnovsky and P. Seymour, "Recognizing Berge graphs," Technical report, Princeton University, 2003.
12. G. Cornuejols, X. Liu, and K. Vuskovic, "A polynomial algorithm for recognizing perfect graphs," Technical report, Carnegie Mellon University, 2003.
13. U. Feige, "Randomized graph products, chromatic numbers, and the Lovász's theta function," Combinatorica, vol. 17, no. 1, pp. 79–90, 1997.
14. M. Grötschel, L. Lovász, and A. Schrijver, "Polynomial algorithms for perfect graphs," Annals of Discrete Mathematics, pp. 325–356, 1984.
15. M. Grötschel, L. Lovász, and A. Schrijver, Geometric Algorithms and Combinatorial Optimization, Springer, New York, 1988.
16. G. Gruber and F. Rendl, "Computational experience with stable set relaxations," SIAM Journal on Optimization, vol. 13 no. 4, pp. 1014–1028, 2003.
17. J. Hastad, "Clique is hard to approximate within $n^{1-\epsilon}$," Acta Mathematica, vol. 182, no. 1, pp. 105–142, 1999.
18. C. Helmberg and F. Rendl, "A spectral bundle method for semidefinite programming," SIAM Journal on Optimization, vol. 10, no. 3, pp. 673–696, 2000.
19. S. Hougardy, "Inclusions between classes of perfect graphs," Technical report, Humboldt-Universität zu Berlin, 1998. Available at http://www.informatik.hu-berlin.de/~hougardy/paper/classes.html.
20. D.E. Knuth, "The sandwich theorem," Electronic Journal of Combinatorics, vol. 1, no. 1, A1, pp. 1–48, 1994.
21. L. Lovász, "On the Shannon capacity of a graph," IEEE Transactions on Information Theory, vol. 25, pp. 1–7, 1979.
22. L. Lovász and A. Schrijver, "Cones of matrices and set-functions and 0-1 optimization," SIAM Journal on Optimization, vol. 1, no. 2, pp. 166–190, 1991.
23. J. Mycielski, "Sur le coloriage des graphes," Colloq. Math., vol. 3, 1955.
24. T. Stephen and L. Tunçel, "On a representation of the matching polytope via semidefinite liftings," Mathematics of Operations Research, vol. 24 no. 1, pp. 1–7, 1999.
25. R.H. Tütüncü, K.C. Toh, and M.J. Todd, "Solving semidefinite-quadratic-linear programs using SDPT3," Mathematical Programming, vol. 95, pp. 189–217, 2003.