ROUNDING THE LOVÁSZ THETA FUNCTION WITH A VALUE FUNCTION APPROXIMATION

Rui Gong[*] **Diego Cifuentes**[†] **Alejandro Toriello**[†]

April 27, 2026

ABSTRACT

The Lovász theta function is a semidefinite programming (SDP) relaxation for the maximum weighted stable set problem, which is tight for perfect graphs. However, even for perfect graphs, there is no known rounding method guaranteed to extract a maximum weighted stable set from the SDP solution. In this paper, we develop a novel rounding scheme for the theta function that constructs a value function approximation from the SDP solution and then generates a stable set using dynamic programming. Our method provably recovers a maximum weighted stable set in several sub-classes of perfect graphs, including generalized split graphs, which asymptotically cover almost all perfect graphs. To the best of our knowledge, this is the only known rounding strategy for the theta function that recovers a maximum weighted stable set for large classes of perfect graphs. Our rounding scheme relies on simple linear algebra computations; we only solve one SDP. In contrast, existing methods that leverage the theta function require solving multiple SDPs. Computational experiments show that our method produces good solutions even on imperfect graphs.

***K*eywords** stable set · Lovász theta function · value function approximation · semidefinite program · perfect graph.

1 Introduction

Semidefinite programming (SDP) relaxations provide a tractable approach for tackling hard combinatorial optimization problems. Two of the most studied cases are SDP relaxations for the *maximum weighted stable set (MWSS) problem* , known as the *Lovász theta function*, and for the maximum cut problem. If the relaxation is tight and the optimal solution has rank one, it is easy to recover a solution of the combinatorial problem from the SDP solution. However, if the optimal solution has higher rank, a *rounding* procedure may be needed, even if the upper bound from the relaxation is tight. Devising good rounding procedures is hence of great importance both in theory and in practice. For example, Goemans and Williamson [27] introduced a randomized rounding algorithm for the maximum cut SDP, which produces a solution with an approximation factor of roughly 0.878. While some heuristic rounding strategies have been proposed for MWSS problem using the theta function [3,57], none of them have theoretical guarantees. In this paper, we introduce a rounding scheme for this SDP that provably finds an MWSS in several important subclasses of perfect graphs, which asymptotically include almost all perfect graphs. Moreover, our rounding scheme performs well in practice, even in imperfect graphs.

Given a simple graph $G = (N, E)$ and a weight function $w : N \to \mathbb{R}_{++}$, the MWSS problem seeks the stable set $S \subseteq N$ that maximizes $\sum_{i \in S} w_i$; a set S is *stable* (or *independent* or a *packing*) if $ij \notin E$ for every pair of vertices $i, j \in S$. The stability number $\alpha(G; w)$ is the optimal value of the problem; computing $\alpha(G; w)$ is NP-hard for general graphs [21,34]. Lovász introduced a quantity $\vartheta(G; w)$, the theta function, which upper bounds $\alpha(G; w)$ and can be efficiently computed by solving an SDP [28, 38]. Indeed, $\vartheta(G; w)$ is the optimal value of the following primal-dual pair of SDPs [28–30]:

[*]{rgong44,diego.cifuentes,atoriello}@gatech.edu
Georgia Institute of Technology, Atlanta, GA 30332, USA
The authors' work was partially funded by the U.S. Office of Naval Research, N00014-23-1-2631.

$$\begin{aligned} \max \quad & w^\top x \\ \text{s.t.} \quad & X_{ii} = x_i, \forall i \in N \\ & X_{ij} = 0, \forall ij \in E \\ & X - xx^\top \succeq 0 \end{aligned} \qquad \text{(SDP-P)} \qquad \begin{aligned} \min \quad & t \\ \text{s.t.} \quad & Q_{ii} = -2q_i - w_i, \forall i \in N \\ & Q_{ij} = 0, \forall ij \notin E \\ & tQ - qq^\top \succeq 0. \end{aligned} \qquad \text{(SDP-D)}$$

Our rounding method relies on an optimal solution of the primal-dual pair above.

The MWSS problem can be solved in polynomial time for an important class of graphs, the *perfect* graphs; this follows because $\alpha(G;w) = \vartheta(G;w)$ for these graphs [29]. Hence, in a perfect graph we can determine whether a vertex $i \in N$ belongs to an MWSS by checking if $\vartheta(G;w) = \vartheta(G|_{N\setminus i};w)$. We can thus extract an MWSS by solving $\mathcal{O}(n)$ SDPs [30]; see [57] for an improved scheme that uses $\min\{\alpha(G;w), n/3\}$ SDPs. Alternatively, Alizadeh [3] considers a randomized algorithm that requires perturbing the weight vector w up to $\mathcal{O}(\log(1/\varepsilon))$ times in order to guarantee that (SDP-P) has a unique rank-one optimal solution with probability $1-\varepsilon$. All previous polynomial-time methods for the MWSS problem in perfect graphs require solving multiple SDPs. In addition, they do not rely on rounding an SDP solution, but instead either use the SDP optimal value or require an exact, rank-one optimal solution; in contrast, our rounding procedure requires solving (SDP-D) once.

To describe the main idea behind our rounding method, let $(x,X),(t,q,Q)$ be optimal solutions of (SDP-P), (SDP-D) respectively. We consider $\mathscr{V}_{\text{SDP}} : 2^N \to \mathbb{R}_+$ defined as

$$\mathscr{V}_{\text{SDP}}(S) := q_S^\top Q_S^\dagger q_S, \quad S \neq \emptyset, \qquad\qquad \mathscr{V}_{\text{SDP}}(\emptyset) := 0, \tag{1}$$

where q_S and Q_S are respectively the sub-vector of q and principal sub-matrix of Q indexed by S, and $Q_S^\dagger$ denotes the Moore–Penrose pseudo-inverse of Q_S. $\mathscr{V}_{\text{SDP}}$ can be computed either using an eigenvalue decomposition or using equivalent characterizations we introduce in later sections. Our rounding algorithm evaluates $\mathscr{V}_{\text{SDP}}$ at most $\mathcal{O}(n^2)$ times, and produces a stable set $S \subseteq N$ for an arbitrary graph G. Notice that q and Q are fixed and given by solving (SDP-D) once. The function $\mathscr{V}_{\text{SDP}}$ is monotone and satisfies the following important property,

$$\mathscr{V}(S) - \mathscr{V}(S \setminus (\{i\} \cup \delta_i)) \geq w_i, \qquad S \subseteq N, i \in S, \tag{2}$$

where δ_i denotes the set of neighbors of i in G. More generally, we call $\mathscr{V} : 2^N \to \mathbb{R}_+$ a *value function approximation* (VFA) for the MWSS problem if the function is monotone, $\mathscr{V}(\emptyset) = 0$, and it satisfies (2). The term *value function* comes from dynamic programming (DP). If we view the construction of a stable set as a sequential decision process, the *state* is the set of available vertices $S \subseteq N$. Selecting a vertex $i \in S$ yields an immediate reward of w_i, and transitions the state to the remaining available vertices $S \setminus (\{i\} \cup \delta_i)$. In this context, the optimal value function—which represents the maximum possible weight achievable from state S—is exactly the stability number $\mathscr{V}^*(S) := \alpha(G|_S;w)$. By the Bellman equation, this optimal value function naturally satisfies (2).

Our rounding procedure starts by discarding all vertices i with $x_i = 0$. We then arbitrarily select a remaining vertex to be included in the stable set and discard its neighbors. We keep discarding vertices with $\mathscr{V}_{\text{SDP}}(S) - \mathscr{V}_{\text{SDP}}(S \setminus (\{i\} \cup \delta_i)) > w_i$ and selecting until no vertices are left, and then return the selected set of vertices. This process may sometimes lead to choosing a vertex incorrectly, so we use a DP technique known as a one-step *look-ahead* to prevent this from occurring: we simulate a vertex choice and check if it leads to a suboptimal stable set; if so, we backtrack and delete that vertex. The output of our procedure is always a stable set; we show that it is indeed an MWSS for several important subclasses of perfect graphs.

The analysis of our rounding scheme uses the clique linear programming (LP) relaxation of the MWSS problem. Consider the primal-dual pair of LPs

$$\begin{aligned} \max_{x \geq 0} \quad & w^\top x \\ \text{s.t.} \quad & \sum_{i \in C} x_i \leq 1, \forall C \in \mathscr{C}(G), \end{aligned} \qquad \text{(LP-P)} \qquad \begin{aligned} \min_{\mu \geq 0} \quad & \sum_{C \in \mathscr{C}(G)} \mu_C \\ \text{s.t.} \quad & \sum_{C \ni i} \mu_C \geq w_i, \forall i \in N, \end{aligned} \qquad \text{(LP-D)}$$

where $\mathscr{C}(G)$ is the set of all cliques in G; this relaxation is tight and the primal polyhedron of (LP-P) is integral for perfect graphs [29]. Given a solution of (LP-D), we can construct another VFA $\mathscr{V}_{\text{LP}} : 2^N \to \mathbb{R}_+$ and apply our algorithm with $\mathscr{V}_{\text{LP}}$. We emphasize that our algorithm does not solve (LP-D), but we use it to analyze our rounding scheme with (SDP-D). Our main results are stated next.

Theorem 1.1 (Informal)**.** *Our rounding algorithm based on either* (LP-D) *or* (SDP-D) *outputs a maximum weighted stable set for generalized split, chordal, and co-chordal graphs.*

The precise statement of our main theorem is given in Section 2. Note that almost all perfect graphs are generalized split graphs in an asymptotic sense [52]. To the best of our knowledge, our methods give the only known rounding strategy for the Lovász theta function that provably works for large subclasses of perfect graphs.

The rest of the paper is organized as follows. Section 2 provides background, properties of a VFA, our algorithm and the formal statement of the main results. Section 3 analyzes our algorithm with $\mathscr{V}_{\text{LP}}$, discusses its combinatorial interpretation, and highlights the necessity of the look-ahead. Then Section 4 presents the analysis of our algorithm with $\mathscr{V}_{\text{SDP}}$ and proves the main results. Section 5 describes our computational study.

1.1 Brief Literature Review

Beyond the work mentioned above, there is a huge body of literature on the MWSS problem. While it is NP-hard for general graphs [21, 34], there are polynomial-time combinatorial algorithms for various graph classes, including circular graphs and their complements [23], circular arc graphs and their complements [24, 31], graphs without long odd cycles [32], claw-free graphs and ℓ-claw-free graphs [17, 18], and graphs without two disjoint odd cycles [15]. For graphs without holes of length at least five, the results in [1] imply an $n^{\mathscr{O}(k)}$ algorithm, where k is an upper bound for the treewidth. Recently, [54] gave a fixed-parameter-tractable algorithm for the cardinality (unweighted) objective that depends exponentially on $g := (d+1) - \alpha$, where d, α are respectively the degeneracy and unweighted stability number. There are polynomial-time combinatorial algorithms for the cardinality objective on some sub-classes of perfect graphs, including chordal graphs [25] and generalized split graphs [16], which we also study. Unlike these works, which are tailored to a graph class, our algorithm is generic, works with arbitrary weights, and can be applied to any perfect graph; it furthermore exhibits favorable computational performance. To the best of our knowledge, there is no known polynomial-time combinatorial algorithm for general perfect graphs.

There are also exact approaches for the MWSS problem based on integer programming techniques. These include branch-and-bound [5, 6, 9, 10, 13, 40, 47, 50], which often derive bounds from clique covers, and cutting-plane algorithms, e.g. [8, 46, 47]. There are also highly effective heuristic and meta-heuristic methods for the stable set problem, such as tabu search [19]. The algorithm proposed in [12] is particularly relevant for us, as it uses (SDP-P); despite having no theoretical guarantees, it performs quite well in our computational experiments. For further study on relevant algorithms, we refer the reader to [41, 55] and the references therein.

Our algorithm can be adapted to efficiently output all maximum weighted stable sets (rather than one of them) for unipolar, chordal and co-chordal graphs. Algorithms that enumerate all optimal solutions of a problem have a long history in combinatorial optimization, beginning with ranked-enumeration frameworks for assignments [45], and the K-best procedure [35], which shows how to systematically recover an entire family of optimal solutions rather than a single optimum. For stable sets, however, all-optima results are much rarer than single-solution algorithms to our knowledge: [49] show that all maximum stable sets of a chordal graph can be enumerated in constant amortized time per output, and [37] consider the problem of finding all maximum weighted stable sets in interval and circular-arc graphs. By contrast, standard references for unipolar graphs, e.g. [16], focus on computing one maximum stable set efficiently rather than enumerating the full family of optimal solutions.

1.2 Contributions

- We introduce a novel rounding method for extracting a stable set from the optimal solution of the Lovász theta function SDP. Our rounding procedure relies on a VFA constructed from the dual variables.
- Our rounding procedure is guaranteed to extract an MWSS for generalized split, chordal, and co-chordal graphs, and can be modified to efficiently output all MWSS for unipolar, chordal and co-chordal graphs. To our knowledge, this is the first rounding procedure for the theta function SDP that provably works for any of these classes.
- We test our method in several perfect and imperfect graphs with up to 60,000 nodes. The experiments show that our method produces near-optimal solutions and is computationally efficient; the majority of the runtime is taken up by solving the SDP.

1.3 Notation

We let $\mathbb{R}$ be the real numbers. For a finite set N, we denote the set of $N \times 1$ vectors, non-negative vectors and positive vectors by $\mathbb{R}^N$, $\mathbb{R}^N_+$, $\mathbb{R}^N_{++}$ respectively; the sets of $N \times N$ symmetric matrices, positive semidefinite (PSD) matrices, and positive definite matrices are $\mathbb{S}^N$, $\mathbb{S}^N_+$, and $\mathbb{S}^N_{++}$, respectively. For $A, B \in \mathbb{S}^N$, the partial order $A \succeq B$ is defined by $A - B \in \mathbb{S}^N_+$, and similarly, $A \succ B$ is defined by $A - B \in \mathbb{S}^N_{++}$. For $N = [n]$ or when $|N| = n$, we sometimes use $\mathbb{R}^n$ instead of $\mathbb{R}^N$ for convenience, and the same applies to other sets with superscript N. For any $q \in \mathbb{R}^N, Q \in \mathbb{S}^N$ and

$S \subseteq N$, we respectively let q_S, Q_S denote the sub-vector and principal sub-matrix of q, Q indexed by S. Given a matrix $D \in \mathbb{S}^n$, let $\operatorname{diag}(D) = (D_{11}, \ldots, D_{nn}) \in \mathbb{R}^n$ be the vector consisting of its diagonal entries. Conversely, given $d \in \mathbb{R}^n$, let $\operatorname{Diag}(d) \in \mathbb{S}^n$ be the diagonal matrix with d in its diagonal.

Throughout the paper, assume that $G = (N, E)$ is a simple undirected graph, where N is the set of vertices, E is the set of edges, and $n := |N|$, $m := |E|$. Denote the complement of G by $\overline{G}$, $N \setminus S$ by $\overline{S}$, and let $G|_S$ be the induced subgraph of G on $S \subseteq N$. We let $w : N \to \mathbb{R}_{++}$ denote a weight function over N. For a vertex $i \in N$, we let δ_i denote the set of its neighbors in G, and let δ_S denote the set of vertices j such that $j \in \overline{S}$ and $j \in \delta_i$ for some $i \in S$. A set $S \subseteq N$ is *stable* in G if no two vertices in S are adjacent, and is a *clique* if every two vertices in S are adjacent. A stable set (clique) S is a *maximum weighted stable set (clique)* if it maximizes $\sum_{i \in S} w_i$ among all stable sets (cliques) in G. For w and $S \subseteq N$, we let $\alpha(S) := \alpha(G|_S; w)$ denote the weight of a max stable set contained in S.

2 Preliminaries and Algorithm

In this section, we detail necessary background, then introduce our algorithms and some properties of the VFA. Finally, we introduce the VFAs based on LP and SDP, and state the main results.

2.1 Perfect Graphs and Convex Relaxations

A graph G is *perfect* if the chromatic number $\chi(G')$ equals the clique number $\omega(G')$ for every induced subgraph G' of G. Equivalently, G is perfect if and only if it does not contain a chordless odd cycle (an *odd hole*) or its complement (an *odd anti-hole*) as an induced subgraph [14].

The MWSS problem can be formulated as an LP over the *stable set polytope*,

$$\operatorname{STAB}(G) := \operatorname{conv}\{x \in \{0,1\}^N : x \text{ is an incidence vector of a stable set in } G\}.$$

Let $\operatorname{CLQ}(G) \subseteq \mathbb{R}^N$ be the feasible set of (LP-P), and let $\operatorname{TH}(G) \subseteq \mathbb{R}^N$ be the projection of the feasible set of (SDP-P) onto the x variables; $\operatorname{TH}(G)$ is known as the *theta body* of G [39]. The following relations hold [28–30]:

$$\operatorname{STAB}(G) \subseteq \operatorname{TH}(G) \subseteq \operatorname{CLQ}(G); \qquad \operatorname{STAB}(G) = \operatorname{TH}(G) = \operatorname{CLQ}(G), \text{ if } G \text{ is perfect.} \tag{3}$$

Thus, the problems (LP-P) and (SDP-P) are convex relaxations of the MWSS problem, and both relaxations are tight for perfect graphs.

Problem (SDP-P) can be solved in polynomial time using interior point methods [48]. Interior point methods return a point in the relative interior of the *optimal face* of (SDP-P); see, e.g., [4, Theorem 3.7]. We call such a point a *relative interior solution*. This property of interior point methods contrasts with methods such as simplex, which return extreme points of the optimal face.

2.2 Retrieving a Stable Set from a VFA

Let $G = (N, E)$ be a simple undirected graph and $w : N \to \mathbb{R}_{++}$ be a weight function over N. For a set of vertices $I \subseteq N$, we let $\alpha(I)$ denote the weighted stability number of the induced subgraph $G|_I$. Interpreted as a set function $\alpha : 2^N \to \mathbb{R}_+$, it is the optimal value function of the MWSS problem. A VFA is a function $\mathscr{V} : 2^N \to \mathbb{R}_+$ with:

(i) $\mathscr{V}(\emptyset) = 0$.

(ii) $\mathscr{V}$ is monotone, $\mathscr{V}(I) \leq \mathscr{V}(J)$ for $I \subseteq J$.

(iii) $\mathscr{V}$ satisfies (2).

We say a VFA is *tight* if $\mathscr{V}(N) = \alpha(N)$. We now show that a VFA upper bounds the optimal value function, $\alpha(I)$.

Lemma 2.1. *Given a VFA $\mathscr{V}$, $\mathscr{V}(I) \geq \alpha(I)$ for all $I \subseteq N$.*

Proof. For $I = \{1, \ldots, s, s+1, \ldots, |I|\}$, without loss of generality, assume $S^* = \{1, \ldots, s\}$ is an MWSS of $G|_I$. Then by applying (2) repeatedly,

$$\mathscr{V}(I) \geq w_1 + \mathscr{V}(I \setminus (\delta_1 \cup \{1\})) \geq \cdots \geq \sum_{i \in S^*} w_i + \mathscr{V}(I \setminus (\delta_{S^*} \cup S^*)) = \alpha(I).$$

The last inequality follows because S^* is an MWSS of $G|_I$, and thus $I \subseteq (\delta_{S^*} \cup S^*)$. □

The following are sufficient conditions for a set function to be a VFA.

Lemma 2.2. *Let $\mathscr{V} : 2^N \to \mathbb{R}_+$ satisfy (i) $\mathscr{V}(\emptyset) = 0$, (ii) $\mathscr{V}$ is monotone, (iii) $\mathscr{V}(\{i\}) \geq w_i$ for $i \in N$, and (iv) $\mathscr{V}(I \cup J) = \mathscr{V}(J) + \mathscr{V}(I)$ for all disjoint $I, J \subseteq N$ with no edge between them. Then $\mathscr{V}$ is a VFA.*

Proof. Suppose all the conditions above hold. Since there is no edge between i and $J := I \setminus (\{i\} \cup \delta_i)$, by (iv) we have that $\mathscr{V}(i \cup J) = \mathscr{V}(i) + \mathscr{V}(J)$. Using (ii) and (iii) we get that

$$\mathscr{V}(I) \geq \mathscr{V}(i \cup J) = \mathscr{V}(i) + \mathscr{V}(J) \geq w_i + \mathscr{V}(J).$$

Hence, $\mathscr{V}$ is a VFA. □

Algorithm 1 constructs a stable set from any VFA. The algorithm maintains a stable set $S \subseteq N$ and a set $I \subseteq N$ of vertices yet to be processed. When a vertex i is selected to be in S, it is discarded from I together with its neighbors. Algorithm 1 is designed to work with a tight VFA; our main theoretical contribution is the analysis of Algorithm 1 on several classes of perfect graphs, where we can construct tight VFAs because of (3).

Algorithm 1 Retrieving a stable set from a tight VFA with one-step look ahead

Input: $G = (N, E)$, a weight function w, optimal $x^* \in \text{STAB}(G)$, and a tight VFA $\mathscr{V}$.
Phase I:
$S \leftarrow \emptyset$ ▷ Start with the empty stable set.
$I \leftarrow \{i \in N : x^*_i > 0\}$ ▷ Discard vertices not in any optimal set.
Phase II:
while $I \neq \emptyset$ **do**
  $I' \leftarrow I$
  $I' \leftarrow I' \setminus (\{i\} \cup \delta_i)$ for an arbitrary $i \in I'$ ▷ Tentatively select i and discard δ_i.
  while $\exists j \in I'$ with $\mathscr{V}(I') - \mathscr{V}(I' \setminus (\{j\} \cup \delta_j)) > w_j$ **do**
    $I' \leftarrow I' \setminus \{j\}$ ▷ Discard vertices that cannot be in an optimal set with i.
  if $\mathscr{V}(I) > \mathscr{V}(I') + w_i$ **then** ▷ Check if i is a suboptimal choice.
    $I \leftarrow I \setminus \{i\}$
  else
    $I \leftarrow I', S \leftarrow S \cup \{i\}$ ▷ Confirm choice of i and continue.
Output: Return S, a stable set of G.

Definition 2.3. *During each iteration of **Phase II** of Algorithm 1, if the set of selected vertices S is a subset of an MWSS of G, we say we are on an* optimal trajectory. *If an iteration starts with the remaining set of vertices I and $S \cup \{i\}$ is a subset of an MWSS of G for some $i \in I$, then we say i is an* optimal choice *of this iteration or i is* on an optimal trajectory*, otherwise, we say i is* suboptimal *or not on an optimal trajectory.*

After making a copy I' of I and tentatively selecting a vertex i in **Phase II**, Algorithm 1 tentatively discards vertices that are not in any MWSS of the current graph $G|_{I'}$. It then checks if i is indeed an optimal choice: if not, Algorithm 1 returns to I, deletes i, and continues; otherwise, Algorithm 1 adds i to S, updates I as I' and continues. The following lemma justifies these claims.

Lemma 2.4. *Let $I \subseteq N$, $i \in I$. A VFA $\mathscr{V}$ has the following properties:*

(i) If $\mathscr{V}(I) = \alpha(I)$ and $\mathscr{V}(I) - \mathscr{V}(I \setminus (\{i\} \cup \delta_i)) > w_i$, then i is not in any MWSS of $G|_I$.

(ii) If $\mathscr{V}(I) = \alpha(I)$ and either $\mathscr{V}(I) - \mathscr{V}(I \setminus (\{i\} \cup \delta_i)) > w_i$ or $\alpha(I) = \alpha(I \setminus \{i\})$, then $\mathscr{V}(I \setminus \{i\}) = \alpha(I \setminus \{i\})$.

*(iii) Suppose $\mathscr{V}$ is tight. At the start of any iteration of **Phase II** of Algorithm 1, if S is on an optimal trajectory, then $\mathscr{V}(I) = \alpha(I)$ and for any MWSS S_I of $G|_I$, $S \cup S_I$ is an MWSS of G.*

Proof. (i) For contradiction, suppose i is in an MWSS S^* of $G|_I$. Without loss of generality, suppose $i = 1$; by (2),

$$\mathscr{V}(I) > w_1 + \mathscr{V}(I \setminus (\delta_1 \cup \{1\})) \geq \ldots \geq \sum_{i \in S^*} w_i + \mathscr{V}(I \setminus (\delta_{S^*} \cup S^*)) = \sum_{i \in S^*} w_i + 0 = \alpha(I),$$

which contradicts $\mathscr{V}(I) = \alpha(I)$.

(ii) By the first property of this lemma and Lemma 2.1, $\mathscr{V}(I) = \alpha(I) = \alpha(I \setminus \{i\}) \leq \mathscr{V}(I \setminus \{i\})$. Since $\mathscr{V}$ is monotone, $\alpha(I \setminus \{i\}) = \mathscr{V}(I) = \mathscr{V}(I \setminus \{i\})$.

(iii) We proceed by induction. The base case follows from the tightness assumption. Assume the set of selected vertices S is a subset of an MWSS of G, and the remaining set of vertices I satisfies $\mathscr{V}(I) = \alpha(I)$. Suppose $i \in I$ is selected in the current iteration and $S \cup \{i\}$ is a subset of an MWSS of G. Let us show that the statement holds for the next iteration.

We claim that i belongs to an MWSS of $G|_I$. Suppose not; then

$$\alpha(I) > \alpha(I \setminus (\{i\} \cup \delta_i)) + w_i \implies \alpha(N) > \sum_{j \in S \cup \{i\}} w_j + \alpha(I \setminus (\{i\} \cup \delta_i)),$$

so $S \cup \{i\}$ is not a subset of an MWSS of G, a contradiction. Set $I' \leftarrow I \setminus (\{i\} \cup \delta_i)$ and let S_I be an MWSS of $G|_I$ containing i. Then $S_I \setminus \{i\}$ is an MWSS of I' and

$$\mathscr{V}(I') = \mathscr{V}(I \setminus (\{i\} \cup \delta_i)) = \mathscr{V}(I) - w_i = \alpha(I \setminus (\{i\} \cup \delta_i)).$$

After discarding all vertices j with $\mathscr{V}(I') - \mathscr{V}(I' \setminus (\{j\} \cup \delta_j)) > w_j$ and updating I', by the second property of this lemma, $\mathscr{V}(I \setminus (\{i\} \cup \delta_i)) = \mathscr{V}(I')$, which implies $\mathscr{V}(I') = \alpha(I \setminus (\{i\} \cup \delta_i))$. Again by the first property of this lemma, $\alpha(I \setminus (\{i\} \cup \delta_i)) = \alpha(I')$, hence $\mathscr{V}(I') = \alpha(I')$. By the induction assumption and the tightness of $\mathscr{V}$,

$$\mathscr{V}(N) = \sum_{j \in S} w_j + \mathscr{V}(I) = \sum_{j \in S} w_j + w_i + \alpha(I \setminus (\{i\} \cup \delta_i)) = \sum_{j \in S} w_j + w_i + \alpha(I') = \alpha(N).$$

Thus, for any MWSS $S_{I'}$ of $G|_{I'}$, $S_{I'} \cup (S \cup \{i\})$ is an MWSS of G.

□

Corollary 2.5. *Suppose a VFA $\mathscr{V}$ is tight. At the beginning of any iteration of* ***Phase II*** *of Algorithm 1, if S is a subset of an MWSS of G, then $\sum_{i \in S} w_i + \mathscr{V}(I) = \alpha(N)$.*

Lemma 2.4 provides important properties of our algorithm. First, given $\mathscr{V}(I) = \alpha(I)$, discarding vertices with $\mathscr{V}(I) - \mathscr{V}(I \setminus (\{i\} \cup \delta_i)) > w_i$ preserves all MWSS of $G|_I$. When we discard a vertex that is not in any MWSS, the VFA remains equal to the stability number. Also, if our algorithm is following an optimal trajectory at every iteration, $\mathscr{V}(I)$ preserves the stability number of $G|_I$, and Algorithm 1 outputs an MWSS when it terminates. Hence, with any VFA, our goal is to stay on an optimal trajectory.

2.3 VFAs from LP/SDP Relaxations

Consider the primal-dual pair of LP relaxations (LP-P), (LP-D). Given a feasible μ for (LP-D), we define a VFA

$$\mathscr{V}_{\mathrm{LP}}(S) := \sum_{C \in \mathscr{C}(G), C \cap S \neq \emptyset} \mu_C, \quad S \subseteq N. \tag{4}$$

The following lemmas verify that $\mathscr{V}_{\mathrm{LP}}$, $\mathscr{V}_{\mathrm{SDP}}$ are VFAs that are tight when the graph is perfect.

Lemma 2.6. *If μ is feasible for* (LP-D)*, $\mathscr{V}_{LP}$ satisfies the conditions from Lemma 2.2, and hence is a VFA. If μ is optimal for* (LP-D) *and G is perfect, $\mathscr{V}_{LP}$ is a tight VFA.*

Proof. We verify the four conditions from Lemma 2.2. (i) Follows from the definition. (ii) Follows from the definition and the fact that $\mu \geq 0$. (iii) $\mathscr{V}_{\mathrm{LP}}(i) = \sum_{C \ni i} \mu_C$, which is greater than or equal to w_i since μ is feasible to (LP-D). (iv) If I, J share no edges, a clique $C \in \mathscr{C}(G)$ can intersect at most one of I, J. Then $\mathscr{V}_{\mathrm{LP}}(I \cup J) = \mathscr{V}_{\mathrm{LP}}(I) + \mathscr{V}_{\mathrm{LP}}(J)$ by definition. The tightness for perfect graphs follows from (3) and the optimality of μ. □

For the primal-dual pair of SDP relaxations (SDP-P), (SDP-D), given a feasible (t, q, Q) for (SDP-D), consider $\mathscr{V}_{\mathrm{SDP}}$ from (1). The following lemma provides some alternative forms of $\mathscr{V}_{\mathrm{SDP}}$, which are useful for analysis and computation.

Lemma 2.7. *Let $q \in \mathbb{R}^N, Q \in \mathbb{S}^N_+$, and define $\mathscr{V}_{SDP}$ with* (1)*. For $\emptyset \neq S \subseteq N$,*

$$\mathscr{V}_{SDP}(S) \quad := \quad q_S^\top \, Q_S^\dagger \, q_S \quad = \quad \min_{t \in \mathbb{R}} \left\{ t : \begin{pmatrix} t & q_S^\top \\ q_S & Q_S \end{pmatrix} \succeq 0 \right\} \quad = \quad -\min_{y \in \mathbb{R}^S} (y^\top Q_S y - 2 q_S^\top y).$$

Proof. We first consider the SDP formulation. By taking the Schur complement, the PSD constraint is equivalent to $t \geq 0, t Q_S \succeq q_S q_S^\top$. Therefore, the SDP is feasible if and only if $Q_S \succeq 0$ and q_S is in the range of Q_S by [2, Theorem 1]. Assume that this is the case, and we show $t^* := q_S^\top Q_S^\dagger q_S$ is feasible, i.e., $t^* Q_S \succeq q_S q_S^\top$. By applying singular value decomposition and changing the basis, we may assume $Q_S = \mathrm{Diag}(a_1, \ldots, a_r, 0, \ldots, 0)$ with $a_i > 0$ and

$q_S = (b_1, \ldots, b_r, 0, \ldots, 0)$ with $b_i \geq 0$, so q_S lies in the range of Q_S. Then $Q_S^\dagger = \text{Diag}(a_1^{-1}, \ldots, a_r^{-1}, 0, \ldots, 0)$. For any vector $x \in \mathbb{R}^n$, we have,

$$x^\top (t^* Q_S - q_S q_S^\top) x = (q_S^\top Q_S^\dagger q_S)(x^\top Q_S x) - (q_S^\top x)^2 = \Big(\sum_{i=1}^r b_i^2 a_i^{-1}\Big)\Big(\sum_{i=1}^r x_i^2 a_i\Big) - \Big(\sum_{i=1}^r b_i x_i\Big)^2,$$

which is nonnegative by Cauchy-Schwarz inequality. Hence, t^* is feasible. Moreover, when $x = Q_S^\dagger q_S$, the above inequality becomes equality, so t^* is the optimum. Hence, $q_S^\top Q_S^\dagger q_S = \min_{t \in \mathbb{R}} \left\{ t : \begin{pmatrix} t & q_S^\top \\ q_S & Q_S \end{pmatrix} \succeq 0 \right\}$.

For the other equivalence, since Q_S is PSD, the optimum is obtained at y for $Q_S y = q_S$, which is satisfied when $y = Q_S^\dagger q_S$, as we show above. Then $y^\top Q_S y - 2 q_S^\top y = -q_S^\top Q^\dagger q_S$, which implies $-\min_y (y^\top Q_S y - 2 q_S^\top y) = q_S^\top Q_S^\dagger q_S$. □

We proceed to show that $\mathscr{V}_{\text{SDP}}$ is a VFA.

Lemma 2.8. *If (t, q, Q) is feasible for* (SDP-D)*, $\mathscr{V}_{SDP}$ is a VFA. If (t, q, Q) is optimal for* (SDP-D) *and G is perfect, $\mathscr{V}_{SDP}$ is a tight VFA.*

Proof. We proceed to verify the four conditions from Lemma 2.2.

(i) Follows from the definition of $\mathscr{V}_{\text{SDP}}$.

(ii) Let $I \subseteq J$ and let $t_J = \mathscr{V}_{\text{SDP}}(J)$. By the previous lemma, we have that $\begin{pmatrix} t_J & q_J^\top \\ q_J & Q_J \end{pmatrix} \succeq 0$. Since a principal sub-matrix of a PSD matrix is also PSD, we have $\begin{pmatrix} t_J & q_I^\top \\ q_I & Q_I \end{pmatrix} \succeq 0$, and hence $\mathscr{V}_{\text{SDP}}(I) \leq t_J = \mathscr{V}_{\text{SDP}}(J)$ by Lemma 2.7.

(iii) Suppose there is no edge between I, J; then the matrix $Q_{I \cup J}$ is a block diagonal matrix, i.e. $Q_{I \cup J} = \text{Diag}(Q_I, Q_J)$. Then its pseudo-inverse is also block diagonal, $Q_{I \cup J}^\dagger = \text{Diag}(Q_I^\dagger, Q_J^\dagger)$. Hence,

$$\mathscr{V}_{\text{SDP}}(I \cup J) = q_{I \cup J}^\top Q_{I \cup J}^\dagger q_{I \cup J} = q_I^\top Q_I^\dagger q_I + q_J^\top Q_J^\dagger q_J = \mathscr{V}_{\text{SDP}}(I) + \mathscr{V}_{\text{SDP}}(J).$$

(iv) We have $Q_{ii} = -2q_i - w_i$, by feasibility in (SDP-D). If $Q_{ii} = 0$, then again by feasibility, $q_i = 0$, $w_i = -2q_i - Q_{ii} = 0$, and $\mathscr{V}_{\text{SDP}}(i) = 0 \geq w_i = 0$. If $Q_{ii} > 0$, then by the definition,

$$\mathscr{V}_{\text{SDP}}(i) := Q_{ii}^{-1} q_i^2 = \frac{1}{4} Q_{ii}^{-1} (Q_{ii} + w_i)^2 \implies Q_{ii}(\mathscr{V}_{\text{SDP}}(i) - w_i) = \frac{1}{4}(Q_{ii} - w_i)^2 \geq 0,$$

which implies $\mathscr{V}_{\text{SDP}}(i) \geq w_i$.

The tightness follows from (3) and the optimality of (t, q, Q). □

2.4 Statement of main results

Our main contribution is to show that Algorithm 1, applied with either $\mathscr{V}_{\text{LP}}$ or $\mathscr{V}_{\text{SDP}}$, finds an MWSS for several important subclasses of perfect graphs. In Section 3, we provide a combinatorial interpretation of Algorithm 1 with $\mathscr{V}_{\text{LP}}$, to prove the optimality of the returned stable set. Subsequently, in Section 4 we analyze Algorithm 1 with $\mathscr{V}_{\text{SDP}}$, and give a modification that efficiently outputs all MWSS for unipolar, chordal and co-chordal graphs.

Definition 2.9. *A graph G is* unipolar *if its vertex set N can be partitioned as $N = A \cup \overline{A}$, such that the graph $G|_A$, called the* center*, is complete, and the connected components of $G|_{\overline{A}}$, called* clusters*, are also complete. A graph is* co-unipolar *if its complement is unipolar. A* generalized split graph *is a graph that is either unipolar or co-unipolar.*

A graph is chordal *if it does not contain induced cycles of length at least four. A graph is* co-chordal *if its complement is chordal.*

Almost all perfect graphs are generalized split graphs in an asymptotic sense [52]. We now state the main results: Algorithm 1 returns an MWSS for the subclasses of perfect graphs in Definition 2.9, while **Phase II** may need to be run twice only for co-unipolar graphs. The proofs are in Subsection 3.3 and Section 4, respectively.

Theorem 2.10. *Let x^*, μ^* respectively be optimal solutions of* (LP-P) *and* (LP-D)*, both in the relative interior of the optimal face of their respective feasible regions, and let $\mathscr{V}_{LP}$ be the VFA constructed from μ^* via* (4)*. Consider Algorithm 1 executed with $\mathscr{V}_{LP}$.*

(a) If G is unipolar, chordal, or co-chordal, then Algorithm 1 returns an MWSS.

(b) Assume that G is co-unipolar. Let i, j be any adjacent vertices remaining after ***Phase I****. Then either Algorithm 1 returns an MWSS when* ***Phase II*** *starts by selecting i or it returns an MWSS when* ***Phase II*** *starts by selecting j.*

Theorem 2.11. *Let (x^*, X^*), (t^*, q^*, Q^*) respectively be optimal solutions of* (SDP-P) *and* (SDP-D)*, both in the relative interior of the optimal face of their respective feasible regions, and let $\mathscr{V}_{SDP}$ be the VFA constructed from (t^*, q^*, Q^*) via* (1)*. Then Algorithm 1 executed with $\mathscr{V}_{SDP}$ returns an MWSS under the same assumptions as in Theorem 2.10.*

Corollary 2.12. *Algorithm 1 executes $\mathscr{O}(n^2)$ VFA evaluations. Therefore, under the conditions of Theorem 2.11, it returns an MWSS in polynomial time.*

Proof. In every iteration, $\mathscr{O}(n)$ VFA evaluations are required. In the worst case, each iteration requires a look-ahead, and there can be at most n look-ahead steps. So there are at most $\mathscr{O}(n)$ iterations, and the overall number of VFA evaluations is $\mathscr{O}(n^2)$. □

Above we account for the number of VFA evaluations, and distinguish this complexity from the SDP solve time and the complexity of a single VFA evaluation, as the latter two depend on user choice. In particular, for computing the VFA, Lemma 2.7 gives three equivalent forms for $\mathscr{V}_{\text{SDP}}$ that result in different complexities. For example, computing a pseudo-inverse requires $\mathscr{O}(n^3)$ time. In practice, we observe that applying gradient descent to the quadratic programming form is more efficient, but not as numerically stable.

The assumption that the optimal dual solution is in the relative interior of the optimal face cannot be relaxed; the algorithm may return a suboptimal stable set otherwise. Furthermore, the look-ahead is also necessary to guarantee the algorithm's success. In Section 3, we discuss how Algorithm 1 may fail on the graph in Figure 2a when either of the assumptions fails.

3 Analysis via the LP VFA

In this section, we analyze Algorithm 1 with $\mathscr{V}_{\text{LP}}$. We first give a combinatorial interpretation, which provides intuition about how and why a vertex is discarded. Then we discuss why the look-ahead is necessary, and provide an example. After that, we prove Algorithm 1 with $\mathscr{V}_{\text{LP}}$ returns an MWSS for generalized split, chordal, and co-chordal graphs. Finally, we give another example showing that even with look-ahead, Algorithm 1 may fail to return an optimal set on some perfect graphs.

In this section, for an iteration of **Phase II**, we let S denote the set of selected vertices, I denote the set of remaining vertices at the beginning of the iteration, and I' be the set of vertices before step 9.

3.1 Combinatorial Interpretation

Let $G = (N, E)$ be a perfect graph. By (3), one can obtain the incidence vector of an MWSS by solving for an optimal extreme point of (LP-P). The number of cliques in G may grow exponentially, so it is expensive to solve such LPs directly. Nonetheless, we can still extract a useful combinatorial interpretation.

Let (x^*, μ^*) be a pair of strictly complementary optimal solutions of (LP-P) and (LP-D). Note that $x^* \in \text{STAB}(G)$ since the graph is perfect. Furthermore, strict complementarity for the LP is equivalent to being in the relative interior of the optimal face. Therefore,

$$\sum_{C \ni i} \mu_C^* > w_i \iff x_i^* = 0 \iff i \text{ is not in any MWSS of } G,$$
$$\mu_C^* > 0 \iff \sum_{i \in C} x_i^* = 1 \iff \text{every MWSS of } G \text{ contains a vertex in } C.$$

Recall that **Phase I** discards all vertices i with $x_i^* = 0$. By the first equivalence above, it preserves all MWSS, ensuring that each remaining vertex belongs to at least one MWSS. The second equivalence motivates the following definition.

Definition 3.1. *Given a graph G, let $\mathscr{C}(G)$ be the set of its cliques. For $C \in \mathscr{C}(G)$, if every MWSS of G contains a vertex in C, we say C is an* essential clique *of G or C is* essential*. In addition, if every vertex in C belongs to an MWSS of G, C is* strictly essential*.*

Combining the two equivalences above, we provide a combinatorial interpretation of (2) in terms of essential cliques and MWSS. For $I \subseteq N$, we have

$$\begin{aligned}\mathscr{V}_{\mathrm{LP}}(I) - \mathscr{V}_{\mathrm{LP}}(I \setminus (\{i\} \cup \delta_i)) &= [\mathscr{V}_{\mathrm{LP}}(I \setminus \delta_i) - \mathscr{V}_{\mathrm{LP}}(I \setminus (\{i\} \cup \delta_i))] + [\mathscr{V}_{\mathrm{LP}}(I) - \mathscr{V}_{\mathrm{LP}}(I \setminus \delta_i)] \\ &= \sum_{C \ni i} \mu_C^* + \sum_{C \subseteq \delta_i \cup (N \setminus I), C \cap I \cap \delta_i \neq \emptyset} \mu_C^* \geq w_i + 0.\end{aligned}$$

Moreover, for every $I \subseteq N$,

$$\mathscr{V}_{\mathrm{LP}}(I) - \mathscr{V}_{\mathrm{LP}}(I \setminus (\{i\} \cup \delta_i)) = w_i \iff \begin{array}{c} i \text{ belongs to some MWSS of } G \text{ and} \\ C \text{ is not essential } \forall C \subseteq \delta_i \cup (N \setminus I), C \cap I \cap \delta_i \neq \emptyset. \end{array}$$

Notice that the equivalences only provide direct links to the MWSS of G but not necessaryly $G|_I$. In particular, there may exist i, I with $\mathscr{V}_{\mathrm{LP}}(I) - \mathscr{V}_{\mathrm{LP}}(I \setminus (\{i\} \cup \delta_i)) = w_i$, but where i is not in any MWSS of $G|_I$. Next, we characterize how essential cliques translate to induced subgraphs.

Lemma 3.2. *Let $I \subsetneq N$. Suppose Algorithm 1 is on an optimal trajectory and reaches $G|_I$ during some iteration. For every essential clique C of G, $C \cap I$ is an essential clique of $G|_I$ if it is non-empty.*

In the lemma, it is necessary to assume we are on an optimal trajectory; otherwise, $C \cap I$ might not contain any vertex in an MWSS of $G|_I$.

Proof. Let S be the set of selected vertices by Algorithm 1. Note that $C \cap S = \emptyset$, otherwise $C \cap I = \emptyset$, a contradiction. Since C is essential in G and S is on an optimal trajectory, every MWSS of $G|_I$ should contain a vertex in $C \cap I$ by the third property of Lemma 2.4. □

Recall that after **Phase I** every vertex belongs to an MWSS. Hence, after **Phase I**,

$$\mathscr{V}_{\mathrm{LP}}(I) - \mathscr{V}_{\mathrm{LP}}(I \setminus (\{i\} \cup \delta_i)) > w_i \quad \iff \quad \exists C \text{ essential}, C \subseteq \delta_i \cup (N \setminus I), C \cap I \cap \delta_i \neq \emptyset.$$

For any such C, if one is on an optimal trajectory, $C \cap I \subseteq \delta_i \cap I$ is an essential clique of $G|_I$ by Lemma 3.2, which implies that i does not belong to any MWSS of $G|_I$ and should be discarded. With this interpretation, after **Phase I**, Algorithm 1 with $\mathscr{V}_{\mathrm{LP}}$ "shrinks" essential cliques as it discards and selects vertices.

3.2 Algorithm without Look-Ahead

In this subsection, we specify a graph sub-structure that can cause the algorithm to fail without a look-ahead, and provide an example. We start with some preliminary results.

Lemma 3.3. *Consider an iteration of Algorithm 1 that starts with S and I, where S is on an optimal trajectory. With i being selected, if $\mathscr{V}(I) > \mathscr{V}(I') + w_i$ at step 9, i is not in any MWSS of $G|_I$.*

Proof. Since S is on an optimal trajectory and i was not previously discarded, by Lemma 2.4 we have $\alpha(I) = \mathscr{V}(I) = \mathscr{V}(I \setminus (\{i\} \cup \delta_i)) + w_i > \mathscr{V}(I') + w_i \geq \alpha(I') + w_i$. If $\alpha(I) > \alpha(I \setminus (\{i\} \cup \delta_i)) + w_i$, i is not in any MWSS of $G|_I$. Otherwise, $\alpha(I \setminus (\{i\} \cup \delta_i)) + w_i = \alpha(I) = \mathscr{V}(I \setminus (\{i\} \cup \delta_i)) + w_i$, and then by the first and second property of Lemma 2.4, $\mathscr{V}(I \setminus (\{i\} \cup \delta_i)) = \mathscr{V}(I')$, a contradiction. □

The following proposition provides a necessary and sufficient condition for Algorithm 1 to return an MWSS for any graph G.

Proposition 3.4. *Let G be a graph and $\mathscr{V}$ be a tight VFA. Algorithm 1 returns an MWSS of G if and only if at the end of every iteration of **Phase II** in which we select a vertex i, we have $\mathscr{V}(I) = \mathscr{V}(I') + w_i$.*

Proof. ($\Longleftarrow$) For every iteration, if S is on an optimal trajectory, Lemma 3.3 ensures that the look-ahead only discards suboptimal vertices; we can assume Algorithm 1 terminates after K iterations of **Phase II** without applying the look-ahead. Let I^t be the set of available vertices at the beginning of iteration t, where $I^1 = N$, and without loss of generality, let t be the vertex selected during iteration t. Then $\alpha(N) = \mathscr{V}(I^1) = \mathscr{V}(I^2) + w_1 = \cdots = \mathscr{V}(\emptyset) + w_K + \cdots + w_1$. Since $\{1, \ldots, K\}$ is a stable set, it is an MWSS of G.

($\Longrightarrow$) If $\mathscr{V}(I^t) > \mathscr{V}(I^{t+1}) + w_t$ for some t, then $\alpha(N) > \mathscr{V}(\emptyset) + w_K + \cdots + w_1$, which implies $\{1, \ldots, K\}$ is not an MWSS of G. □

We emphasize that Proposition 3.4 requires the equality to hold for *every iteration* in which we add a vertex to S. Thus, for Algorithm 1 to fail, $\mathscr{V}(I) > \mathscr{V}(I') + w_i$ in some iteration; we call such $i \in I$ a *bad choice* of this iteration or of I. Indeed, there may be an iteration where $\mathscr{V}(I) = \mathscr{V}(I') + w_i$, but $\alpha(I) > \alpha(I') + w_i$, which means i is not an optimal choice but the algorithm cannot detect it. The proposition only guarantees that at some iteration later in the process, all remaining vertices will be bad choices; at that point we realize we made a suboptimal choice, but we will not know which vertex it was.

With respect to $\mathscr{V}_{\mathrm{LP}}$, a bad choice i is equivalent to $\mathscr{V}_{\mathrm{LP}}(I) > \mathscr{V}_{\mathrm{LP}}(I') + w_i = \mathscr{V}_{\mathrm{LP}}(I' \cup \{i\})$. By the definition of $\mathscr{V}_{\mathrm{LP}}$, we have

$$\mathscr{V}_{\mathrm{LP}}(I) - \mathscr{V}_{\mathrm{LP}}(I' \cup \{i\}) = \sum_{C \in \mathscr{C}(G), C \cap (I' \cup \{i\}) = \emptyset, C \cap I \neq \emptyset} \mu_C^* > 0.$$

In other words, there is an essential clique $C \in \mathscr{C}(G)$ with $C \cap (I' \cup \{i\}) = \emptyset$ and $C \cap I \neq \emptyset$. If one is on an optimal trajectory, this implies that $C \cap I$ is an essential clique of $G|_I$ that would be discarded by selecting i. However, $C \cap I \not\subseteq \delta_i \cap I$ (otherwise the algorithm would discard i in the previous iteration). We conclude that for the algorithm to fail, after selecting i and discarding δ_i, every vertex of $C \cap I \setminus (\{i\} \cup \delta_i) \neq \emptyset$ contains an essential clique in its neighbor set (checked in step 19), so C is completely discarded. Next we discuss a necessary sub-structure for such a bad choice to exist within Algorithm 1.

Definition 3.5. *A graph $G = (N, E)$ is a* house *if $N = [2k] \cup \{v\}$, where $G|_{[2k]}$ is an even hole and there are $i, j \in [2k]$ such that $G|_{\{i,j,v\}}$ is a triangle. We call v the* top *of the house.*

Proposition 3.6. *Consider any perfect graph $G = (N, E)$ and $\mathscr{V}_{LP}$ constructed from a strictly complementary solution of* (LP-D)*. If in some iteration of Algorithm 1 there is a bad choice $i \in I$, $G|_I$ contains a house as an induced subgraph.*

Proof. At the beginning of some iteration, let S be the selected set of vertices, I be the set of remaining vertices. For the sake of contradiction, suppose that $v^* \in I$ is a bad choice. Let I' be the set of vertices after selecting v^* and step 19 of Algorithm 1. As discussed above, $\mathscr{V}_{\mathrm{LP}}(I) > \mathscr{V}_{\mathrm{LP}}(I') + w_{v^*}$ means there is some $C \in \mathscr{C}(G)$ with $C \cap (I' \cup \{v^*\}) = \emptyset, C \cap I \neq \emptyset$. By Lemma 3.2, without loss of generality we restrict all the cliques we discuss onto I ($C \leftarrow C \cap I$).

First, C is not contained in the neighbor set of v^*; otherwise, v^* would be discarded in a previous iteration. Also, the vertices in C are discarded sequentially, so at some point in steps 6-19, there is only $i \in C$ remaining and there is an essential clique V_1 contained in the neighbor set of i. However, V_1 is itself not an essential clique of $G|_I$, otherwise i would have been discarded before choosing v^*. That is, there exists an essential clique V_1' of $G|_I$ with $V_1 \subsetneq V_1'$, and all vertices in $V_1' \setminus V_1$ have been discarded. Let $z_1 \in V_1' \setminus V_1$. The argument for i can be applied to z_1; some essential clique V_2 is contained in its neighbor set, and V_2' can be found in the same way. This argument applies repeatedly until we reach some $V_{k+1} := C_1 \setminus \delta_{v^*} \neq \emptyset$, where C_1 is an essential clique of $G|_I$ with $C_1 \cap \delta_{v^*} \neq \emptyset$. Define $S_V := C_1 \cap \delta_{v^*}$. This constitutes a chain with alternating discarded vertices and essential cliques, $P_i := i \to V_1 \to z_1 \to V_2 \to \cdots \to z_k \to V_{k+1} \to S_V$. Intuitively, when v^* is selected, δ_{v^*} is discarded from I, so V_{k+1} becomes an essential clique of $I \setminus (v^* \cup \delta_{v^*})$. Then, vertices like z_k, which contain V_{k+1} in their neighbor sets, are discarded; smaller cliques become essential, and the process repeats until i is discarded.

Moreover, i by itself is not an essential clique of $G|_I$ (otherwise we would discard all of its neighbors and select it). Therefore, there is some $j \in C \setminus \{i\}$; suppose this is the second-to-last vertex in C that is discarded. That is, before we discard j, $\{i, j\}$ is an essential clique. Also, without loss of generality, we assume that before j is discarded, all other vertices in the remaining graph except for i and j do not contain an essential clique in their neighbor set, as the argument depends on the induced subgraph discussed below. The same argument for i can be applied to j to find another alternating chain of discarded vertices and essential cliques, $P_j := j \to U_1 \to y_1 \to U_2 \to \cdots \to y_h \to U_{h+1} \to S_U$, where $U_{h+1} := C_2 \setminus \delta_{v^*} \neq \emptyset$ and C_2 is an essential clique of $G|_I$ with $C_2 \cap \delta_{v^*} \neq \emptyset$. Unlike P_i, which has length at least one, P_j may have length 0, that is, $j \in S_U$ is possible; see Example 3.8 below.

Consider picking a vertex from each of V_ℓ, U_g; then P_i, P_j are even paths, paths with an odd number of edges. Similarly, for any $\ell \geq 1$, $i \to V_1 \to z_1 \to \ldots \to V_\ell$ is an odd path and $i \to V_1 \to z_1 \to \ldots \to z_\ell$ is an even path. The same argument applies to j; see Figure 1 for an example. For convenience, let $z_0 := i$, $y_0 := j$.

Claim. *There is no edge between V_1 and j (respectively, U_1 and i).*

Proof. Since $\{i, j\}$ is essential, $u \in V_1$ being adjacent to j implies that $\{i, j\} \subseteq \delta_u$, so u would be discarded before. A similar argument applies to U_1 and i. □

Claim. *For every pair y_g, z_ℓ with $\max\{g, \ell\} \geq 1$, we may assume without loss of generality that there is no edge between them. The same applies to any $u_g \in U_g, v_\ell \in V_\ell$.*

Proof. Suppose not; then we replace j by y_g and i by z_ℓ or by u_g, v_ℓ respectively, until no such pair exists. That is, P_i, P_j become $P_i = z_\ell \to V_{\ell+1} \to \cdots \to z_k \to V_{k+1} \to S_V$; $P_j = y_g \to U_{g+1} \to \cdots \to y_h \to U_{h+1} \to S_U$ or, $P_i = v_\ell \to z_\ell \to \cdots \to z_k \to V_{k+1} \to S_V$; $P_j = u_g \to y_g \to \cdots \to y_h \to U_{h+1} \to S_U$ respectively. So the paths P_i, P_j are either both odd or even. Then the closed walk $v^* \to S_V \to v_{k+1} \to \cdots z_\ell \to y_g \to \cdots u_{h+1} \to S_U \to v^*$ is odd. For the rest of the proof, all we need is that no such adjacent pair exists except for z_ℓ, y_g; it is not required that z_ℓ, y_g are essential. □

Claim. *Without loss of generality, we can assume that $y_g \notin V_\ell$ for any $g, \ell \geq 1$. The same applies for z_ℓ, U_g.*

Proof. Assume $y_g \in V_\ell$. Then y_g is adjancent to $z_{\ell-1}$, so the previous claim applies. □

If for some $g, \ell \geq 0$, $y_g = z_\ell$ or $U_g \cap V_\ell \neq \emptyset$, or there is an edge between y_g and V_ℓ, or between z_ℓ and U_g, then we call such a pair (y_g, z_ℓ), (U_g, V_ℓ), (y_g, V_ℓ), (z_ℓ, U_g) a *knot*. We call the knot with $\min\{g, \ell\}$ minimized the *first knot*.

Claim. *Applying the claims above, $G|_I$ either contains a house as an induced subgraph or does not have a knot.*

Proof. Suppose there is a knot. We consider three cases for the first knot:

1. The first knot (y_g, z_ℓ) with $y_g = z_\ell$ has $\min\{g, \ell\} \geq 1$; then

   $$y_0 \to u_1 \to y_1 \to \ldots \to y_g = z_\ell \to v_\ell \to z_{\ell-1} \to v_{\ell-1} \to \cdots \to v_1 \to z_0 \to y_0$$

   has $2g + 2\ell + 1$ edges and forms an odd hole; or $v_\ell \to z_\ell = y_g \to u_g \to v_\ell$ forms a triangle when i, j are replaced by v_ℓ, u_g. Since the graph is perfect, there is no odd hole, so we consider the second case.

   Consider $z_{\ell-1}, y_{g-1}$, suppose there is no edge between $z_{\ell-1}$, u_g and between y_{g-1}, v_ℓ. Then if $\{z_{\ell-1}, y_{g-1}\} \in E$, $G|_{\{z_{\ell-1}, v_\ell, z_\ell = y_g, u_g, y_{g-1}\}}$ is an induced house. If not, consider $v_{\ell-1} \in V_{\ell-1}, u_{g-2} \in U_{g-2}$. Suppose they are adjacent, if there is no edge between $v_{\ell-1}, y_{g-1}$ or between $u_{g-1}, z_{\ell-1}$, $G|_{\{v_{\ell-1}, z_{\ell-1}, v_\ell, z_\ell = y_g, u_g, y_{g-1}, u_{g-1}\}}$ is an induced house; or we find a triangle and apply the above argument again. Suppose there is no edge between $V_{\ell-1}$ and U_{g-1}; consider $z_{\ell-2}, u_{g-2}$ and apply the same argument, until we either find an induced house or reach i, j. Then since there is no edge between i, U_1 and between j, V_1, $\{i, \ldots, v_{\ell-1}, y_g, u_g, \ldots, j\}$ is an induced subgraph.

   If such edges exist, say $\{z_{\ell-1}, u_g\} \in E$, then if $\{z_{\ell-1}, y_{g-1}\} \in E$, $\{z_{\ell-1}, u_g, y_{g-1}\}$ forms a triangle, apply the above argument to it; otherwise, consider $\{u_g, y_{g-1}, u_{g-1}, v_{\ell-1}, z_{\ell-1}\}$ for some $v_{\ell-1} \in V_{\ell_1}$, $u_{g-1} \in U_{g-1}$. If $\{v_{\ell-1}, u_{g-1}\} \in E$, then to avoid odd holes, either $\{z_{\ell-1}, u_{g-1}\} \in E$ or $\{v_{\ell-1}, y_{g-1}\} \in E$; without loss of generality, consider the first case: then $\{z_{\ell-1}, u_{g-1}, v_{\ell-1}\}$ forms a triangle, and the same argument above applies. If $\{v_{\ell-1}, u_{g-1}\} \notin E$, then consider $z_{\ell-2}, u_{g-2}$ and apply the same argument until reaching $\{i, j\} \in E$; then $\{i, \ldots, z_{\ell-1}, u_g, u_{g-1}, \ldots, j\}$ is an odd hole, contradiction.

2. The first knot (U_g, V_ℓ) has $u^* \in U_g \cap V_\ell$. Then the same argument from the previous case applies by replacing $z_\ell = y_g$ by u^*.

3. The first knot is (y_g, V_ℓ) or (z_ℓ, U_g). Then the same argument from the first case applies by replacing $z_\ell = y_g$ by y_g (or z_ℓ respectively). □

□

Now suppose $G|_I$ does not have a knot. Consider

$$\mathscr{H} := v^* \to S_V \to V_{k+1} \to z_k \to \cdots \to z_0 \to y_0 \to \cdots \to y_h \to U_{h+1} \to S_U \to v^*,$$

an odd cycle with at least five edges; since the graph is perfect, this cycle has a chord. Chords separate the odd cycle into induced odd holes, triangles, and even holes. However, since there is no edge between V_1, j and between U_1, i, there is always a hole in $\mathscr{H}$. Since the graph is perfect, it is an even hole. And since $\mathscr{H}$ is odd, there is always an induced triangle. That is, there is always an induced subgraph of $\mathscr{H}$ which is a house. □

Proposition 3.6 implies that Algorithm 1 with $\mathscr{V}_{\text{LP}}$ returns an MWSS on house-free perfect graphs without applying the look-ahead. And Proposition 3.4 provides a necessary condition for the algorithm to possibly fail: in some iteration, $\mathscr{V}(I) > \mathscr{V}(I') + w_i$. As we indicate above, $\mathscr{V}_{\text{LP}}(I) > \mathscr{V}_{\text{LP}}(I') + w_i$ is equivalent to discarding all vertices of an essential clique. A house like the one in Figure 1 is required for that to happen.

Corollary 3.7. *Consider a triangle-free perfect graph (i.e. a bipartite graph) $G = (N, E)$, and $\mathscr{V}_{LP}$ constructed from a strictly complementary solution of* (LP-D)*. Algorithm 1 returns an MWSS of G without the look-ahead.*

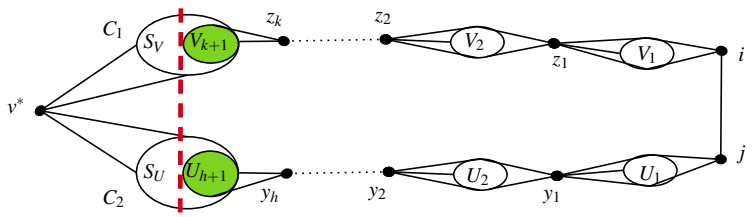

Figure 1: Paths from i, j to v^*.

More generally, in the proof of Proposition 3.6, the only perfect graph property we use is the fact that it does not contain an odd hole. Therefore, Algorithm 1 is guaranteed to return an MWSS if $\mathcal{V}_{\text{LP}}$ is tight and the graph does not contain a house or an odd hole.

For house-free graphs, since there is no bad choice for each iteration, Algorithm 1 returns an MWSS without the look-ahead. However, without the look-ahead, Algorithm 1 can indeed fail to produce an MWSS if a perfect graph contains a house; the following example discusses this in more detail. However, we do not currently know whether the look-ahead is in fact necessary for generalized split, chordal or co-chordal graphs, as the instance in the example is not in any of these families.

Example 3.8. *Consider the perfect graph G in Figure 2a with a cardinality objective, where the yellow cliques with $\mu^*_C = 1$ are the essential cliques determined by a strictly complementary solution of* (LP-D)*; in this case, μ^* is an extreme point and the unique dual optimal solution. We show that this graph is perfect but not generalized split, chordal or co-chordal at the end of this section in Proposition 3.11. Assume Algorithm 1 omits the look-ahead (steps 9-12). Suppose vertex* 10 *is selected; then its neighbors* $9, 11$ *are discarded (see Figure 2b). Then vertex* 8 *becomes essential, so* 7 *is discarded and* $\{3,4\}$ *becomes essential. The resulting graph is shown in Figure 2c, where the essential clique* $\{1,2\}$ *is not strictly essential. Without the look-ahead, the algorithm can't discard any remaining vertex; suppose* 2 *is selected and its neighbors* $1,3,6$ *are discarded. Vertices* $4,5$ *become essential and are also each others' neighbors; see Figure 2d. One of the vertices must be discarded; if* 5 *is discarded, the algorithm returns the stable set* $S = \{2,4,8,10\}$*. However,* $\alpha = 5$*; there are* 5 *essential cliques in G, but S does not contain a vertex from* $\{5,6\}$*.*

Next, suppose we use the look-ahead; for the iteration starting at Figure 2c, $I = \{1,\dots,6\}$*,* 2 *is selected, and* $I' = \{4\}$*, as in Figure 2d. Then we have* $3 = \mathcal{V}_{LP}(I) > \mathcal{V}_{LP}(I') + w_2 = 2$*, so Algorithm 1 goes back to I and discards* 2*, as shown in Figure 2e; Algorithm 1 can select an arbitrary vertex, say* 3*, and return an MWSS* $\{1,3,5,8,10\}$*, as shown in Figure 2f.*

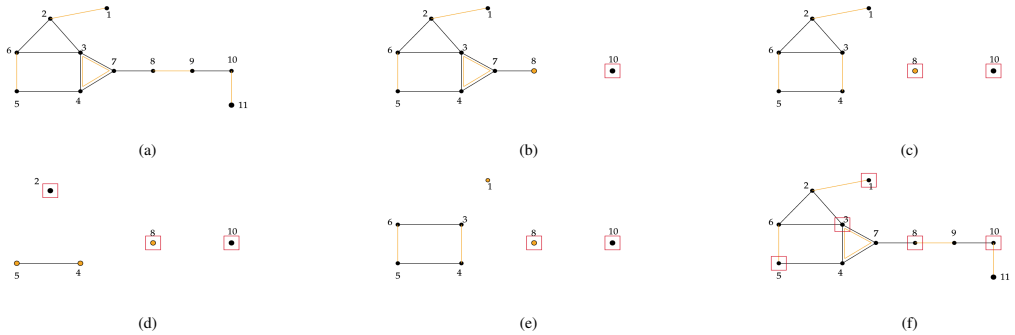

Figure 2

In this example, the sub-graph induced by $\{1,\dots,6\}$ *is a house, and* 2 *is a bad choice. In particular,* 2 *and* 10 *cannot be in the same MWSS, so* 2 *must be discarded once* 10 *is selected. In the next section, we show that Algorithm 1 with $\mathcal{V}_{SDP}$ can discard any vertex that the algorithm discards with $\mathcal{V}_{LP}$ while on an optimal trajectory; therefore, a bad house is also necessary for it to fail. Indeed, Algorithm 1 using $\mathcal{V}_{SDP}$ without the look-ahead can also fail on this instance, by selecting* 2 *after* 10*.*

The example also highlights the importance of using a solution from the relative interior of the optimal face to construct $\mathcal{V}_{LP}$. The three cliques identified in Figure 2c are indeed essential for the sub-graph induced by $\{1,\dots,6\}$*, but they aren't the only essential cliques. The optimal dual solution that assigns unit weight to each of those cliques is optimal for this sub-graph, but it is an extreme point of* (LP-D) *and not in the relative interior of the optimal face.*

As indicated in Example 3.8, Lemma 3.3 and the proof of Proposition 3.4, when $\alpha(I) > \alpha(I') + w_i$, i is not on an optimal trajectory and should be discarded. In the following subsection we show that, for generalized split, chordal and co-chordal graphs, Algorithm 1 can detect and discard any such bad choice.

3.3 Look-Ahead

We next prove Theorem 2.10, which guarantees the algorithm's performance on generalized split, chordal and co-chordal graphs. For co-unipolar graphs, Theorem 2.10 requires the algorithm to be run at most twice, starting from two neighbors. Intuitively, this ensures that at least one run of the algorithm starts from a member of one of the graph's clusters (and not its center). This ensures we obtain an MWSS, because after selecting this vertex, the remaining graph is bipartite, and we apply Proposition 3.6.

Proof of Theorem 2.10(b). Consider G after **Phase I**, which is still co-unipolar. For a co-unipolar graph G, we can separate G into a center A and clusters $B_1, \ldots, B_k$. Let $S = \{v_1, \ldots, v_t\}$ be the returned stable set and v_i be the i-th selected vertex. Without loss of generality, suppose $v_1 \in B_1$; then all $B_2, \ldots, B_k$ and some vertices in A are discarded, so $G|_{N \setminus (v_1 \cup \delta_{v_1})}$ is a bipartite graph and S is an MWSS by Proposition 3.6.

Now suppose $v_1 \in A$; without loss of generality, assume it has at least one neighbor u, which is in $B_1 \cup \ldots \cup B_k$. If S is an MWSS, we are done; otherwise, we run **Phase II** again, and select $u \in B_1 \cup \ldots \cup B_k$ in the first iteration. Then we are back in the previous case. □

For unipolar graphs, we apply the next lemma, which shows that in every iteration of **Phase II** of Algorithm 1, we can stay on an optimal trajectory.

Lemma 3.9. *Let G be a unipolar graph. For each iteration of **Phase II** of Algorithm 1 starting with the set of vertices I, there exists a vertex $u \in I$ such that $\mathscr{V}_{LP}(I) = \mathscr{V}_{LP}(I') + w_u$, where I' is the set of remaining vertices after selecting u and step 19.*

Proof. Consider G after **Phase I**; every vertex in G belongs to an MWSS and $\mathscr{V}_{\text{LP}}$ is constructed from a strictly complementary optimal solution, so the first iteration of **Phase II** does not use the look-ahead. Let A_N be the center and $B_N^1, \ldots, B_N^k$ be the clusters of G. If in every iteration of **Phase II** we have $\mathscr{V}_{\text{LP}}(N) = \alpha(N) = \mathscr{V}_{\text{LP}}(I) + \sum_{i \in S} w_i$, then an MWSS is returned.

For the sake of contradiction, suppose that in some iteration starting with vertex set I, we have $\mathscr{V}_{\text{LP}}(I) > \mathscr{V}_{LP}(I') + w_u$ for every $u \in I$, where I' is the set of vertices remaining after u is selected and vertices are discarded in step 19. Let $A_I \subseteq A_N, B_I^1 \subseteq B_N^1, B_I^2, \subseteq B_N^2, \ldots, B_I^k \subseteq B_N^k$ be the center and clusters of $G|_I$.

Claim. *If $u \in A_I$, then $\mathscr{V}_{LP}(I) = \mathscr{V}_{LP}(I') + w_u$.*

Proof. Suppose there is some $u \in A_I$ with $\mathscr{V}_{\text{LP}}(I) > \mathscr{V}_{\text{LP}}(I') + w_u$. Since u has not been discarded, $\mathscr{V}_{\text{LP}}(I) = \mathscr{V}_{\text{LP}}(I \setminus (u \cup \delta_u)) + w_u$. In other words, $\mathscr{V}_{\text{LP}}(I \setminus (u \cup \delta_u)) > \mathscr{V}_{\text{LP}}(I')$, which implies that some essential clique of I in $I \setminus (u \cup \delta_u \cup I')$ is discarded. That is, after u is selected and δ_u is discarded, there are two disjoint essential cliques, where each is contained in the other's neighbor set, and the algorithm discards one.

Let C_1, C_2 be these two essential cliques with $C_1 \cap C_2 = \emptyset$. Since $u \in A_I$, C_1, C_2 are not in A_I and they belong to the same cluster, say B_I^1. Neither is an essential clique of $G|_I$ nor an essential clique of G contained in B_N^1, otherwise, the other would have been previously discarded. That is, there are two strict essential cliques K_1, K_2 of G such that $C_1 \subsetneq K_1, C_2 \subsetneq K_2$ and $u \notin K_1 \cup K_2$.

Suppose u is selected in the first iteration from N, and δ_u is discarded. Then consider $C_1' := (B_N^1 \setminus \delta_u) \cap K_1, C_2' := (B_N^1 \setminus \delta_u) \cap K_2$. Without loss of generality, assume that vertices not in $C_1' \cup C_2'$ with an essential clique in their neighbor sets have been discarded. If $C_1' \cap C_2' = \emptyset$, these two essential cliques are each contained in the other's neighbor set, so $\alpha(N) > \alpha(N \setminus (u \cup \delta_u)) + w_u$, which implies u is not an optimal choice, and contradicts the assumption that u belongs to an MWSS. Hence, we assume there exists $v \in C_1' \cap C_2'$, which also implies $v \notin \delta_u$. However, $v \notin C_1 \cap C_2 = \emptyset$. If $v \in C_1$, since $\{v\} \cup C_2 \subseteq K_2$, $\{v\} \cup C_2$ is an essential clique instead of C_2 and we have a contradiction, so $v \notin C_1$; similarly, $v \notin C_2$.

Hence, before C_1, C_2 become essential, v is discarded, so it contains an essential clique C in its neighbor. Assume v is discarded in the iteration starting with J, where $J \supseteq I$. We consider three cases:

Case 1: $C \subseteq A_J$: since $u \in A_J$, either $u \in C$ or $C \subseteq \delta_u$. For the latter case, u is discarded in iteration J, so $u \notin I$, a contradiction. If $u \in C$, since $C \subseteq \delta_v$ then $v \in \delta_u$, a contradiction.

Case 2: $C \subseteq A_J \cup B_J^1$ with $C \cap A_J, C \cap B_J^1 \neq \emptyset$: there exists a strictly essential clique K in $A_N \cup B_N^1$ such that $K \supseteq C$ and $v \notin K$. Then $K \setminus \delta_u \neq \emptyset$, otherwise u is not in any MWSS. Also, $u \notin K$; otherwise, since $u \notin C \subseteq \delta_v$, $C \subseteq K \setminus \{u\} \subseteq \delta_u$, so either C is discarded when u is selected from I then v is not discarded, or u is discarded together with v during iteration starting with J, a contradiction. Let $C' := (B_N^1 \setminus \delta_u) \cap K \subseteq \delta_v$ by $v \in B_N^1$. By the properties from above, $C' \neq \emptyset$ and C' is an essential clique after u is selected in the first iteration, so v is discarded.

Case 3: $C \subseteq B_J^1$: there exists a strictly essential clique K in $A_N \cup B_N^1$ such that $K \supseteq C$, $v \notin K$. If $K \cap A_N \neq \emptyset$, then this case is equivalent to the previous one. Suppose $K \subseteq B_N^1$; then since $v \in B_N^1$, we have $K \subseteq \delta_v$, v would be discarded in **Phase I**, and $v \notin C_1' \cup C_2'$, a contradiction. By the three cases above, v is discarded before C_1', C_2' are each contained in the other's neighbor set. Since $v \in C_1' \cap C_2'$ is an arbitrary vertex, we can assume $C_1' \cap C_2' = \emptyset$. Hence, $\mathscr{V}_{\text{LP}}(N) > \mathscr{V}_{\text{LP}}(N \setminus (u \cup \delta_u)) + w_u$ and u is not an optimal choice, a contradiction. □

If every vertex u in I has $\mathscr{V}_{\text{LP}}(I) > \mathscr{V}_{\text{LP}}(I') + w_u$, the claim implies $A_I = \emptyset$. That is, for $u \in B_I^i \neq \emptyset$, we have $\mathscr{V}_{\text{LP}}(I) > \mathscr{V}_{\text{LP}}(I') + w_u$. However, since $A_I = \emptyset$, $I' = I \setminus (\delta_u \cup u) = I \setminus B_I^i$. Since u was not previously discarded, $\mathscr{V}_{\text{LP}}(I) = \mathscr{V}_{\text{LP}}(I \setminus (\delta_u \cup u)) + w_u = \mathscr{V}_{\text{LP}}(I') + w_u$, a contradiction. Hence, for each iteration, there is always a vertex u with $\mathscr{V}_{\text{LP}}(I) = \mathscr{V}_{\text{LP}}(I') + w_u$. □

For chordal graphs, we can again apply the proof of Proposition 3.6 to conclude that in such graphs we do not encounter a house. Finally, for a co-chordal graph, we use the *perfect elimination ordering* [20, page 851] of its complement to show that the algorithm returns an MWSS.

Proof of Theorem 2.10(a). Consider first the case that G is unipolar. Suppose we are on an optimal trajectory and reach I. If $u \in I$ gives $\mathscr{V}_{\text{LP}}(I) > \mathscr{V}_{\text{LP}}(I') + w_u$, then u is not optimal and the look-ahead discards this vertex. If a vertex u is not optimal and not discarded by the look-ahead, then $\alpha(I) = \mathscr{V}_{\text{LP}}(I) = \mathscr{V}_{\text{LP}}(I') + w_u > \alpha(I') + w_u$. For any later iteration, suppose we have vertex set J remaining. If there exists $v \in J$ with $\mathscr{V}_{\text{LP}}(J) = \mathscr{V}_{\text{LP}}(J') + w_v$, select it. Because u is a suboptimal choice, in some iteration every vertex $v \in J$ will have $\mathscr{V}_{\text{LP}}(J) > \mathscr{V}_{\text{LP}}(J') + w_v$, which contradicts Lemma 3.9. Hence, if a suboptimal vertex is selected, the look-ahead always detects and discards it, so Algorithm 1 always returns an MWSS for G by Proposition 3.4.

Suppose next that G is a chordal graph. By definition, it does not contain a house, since it cannot contain an induced cycle of length four or more. Then the proof of Proposition 3.6 implies that Algorithm 1 has $\mathscr{V}_{\text{LP}}(I) = \mathscr{V}_{\text{LP}}(I') + w_u$ for every iteration and every valid choice u. Hence it returns an MWSS.

Finally, suppose G is a co-chordal graph. For the sake of contradiction, assume Algorithm 1 fails on G. That is, during some iteration starting with I, u is selected and

$$\alpha(I) = \mathscr{V}_{\text{LP}}(I) = \mathscr{V}_{\text{LP}}(I') + w_u > \alpha(I') + w_u.$$

As in the proof of Lemma 3.9, we can assume Algorithm 1 keeps picking vertices satisfying $\mathscr{V}_{\text{LP}}(I) = \mathscr{V}_{\text{LP}}(I') + w_u$ until we reach an I where every $u \in I$ has $\mathscr{V}_{\text{LP}}(I) > \mathscr{V}_{\text{LP}}(I') + w_u$. Consider the *perfect elimination ordering* of the chordal graph $\overline{G|_I}$ [20, page 851], and let v be the first vertex in the ordering. For $G|_I$, consider $I' = I \setminus (\delta_v \cup \{v\})$; by the perfect elimination ordering, $\overline{G|_{I'}}$ is a clique and hence $G|_{I'}$ is a stable set. Hence, no neighbors of a vertex in I' contain an essential clique of $G|_{I'}$. Algorithm 1 does not discard any vertex after δ_v is deleted from I and $\mathscr{V}_{\text{LP}}(I) = \mathscr{V}_{\text{LP}}(I') + w_v$, a contradiction. □

Although Theorem 2.10 only applies to certain subclasses of perfect graphs, Algorithm 1 does not require any knowledge about the input graph beyond its perfectness, and it can be applied to any perfect graph. Moreover, in Section 5 we introduce a variant of Algorithm 1 that can be applied to arbitrary, possibly imperfect graphs.

3.4 Look-Ahead May Fail in Some Perfect Graphs

Theorem 2.10 shows that Algorithm 1 returns an MWSS for generalized split, chordal and co-chordal graphs. It is natural to wonder whether Algorithm 1 works for arbitrary perfect graphs. Unfortunately, the answer is negative, as illustrated in the following example.

Example 3.10. *Consider the left perfect graph in Figure 3. Proposition 3.11 below shows that it is perfect but not generalized split, chordal or co-chordal. As before, the yellow cliques represent the essential cliques given by a strictly complementary solution of* (LP-D)*. an MWSS of this graph has size* 9*. As in Example 3.8, suppose* $2, 7, 12$ *are selected; we then obtain the right graph in Figure 3. It is not hard to see that any remaining vertex is a bad choice for the same reason from Example 3.8. Thus, even with look-ahead, Algorithm 1 fails on this graph, and Algorithm 1 with* $\mathscr{V}_{SDP}$ *and look-ahead also fails to obtain an MWSS.*

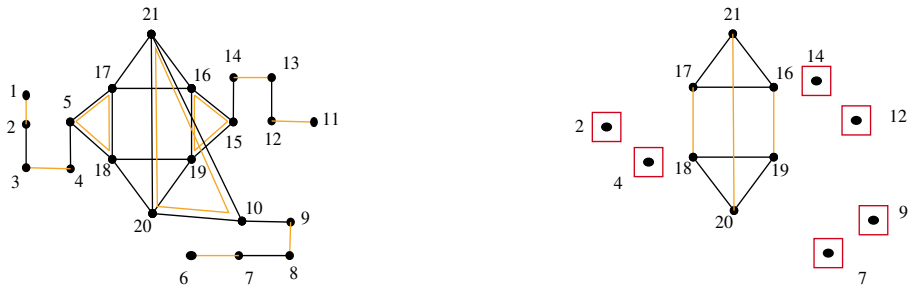

Figure 3: Look-ahead example

By generalizing Example 3.10 with longer paths, Algorithm 1 with a constant number of look-ahead steps can also be made to fail on perfect graphs: In the example, after picking 2 and 7, 12 is a suboptimal choice. By adding steps to the look ahead, we can detect that 14 is suboptimal choice and hence also 12 (after choosing 2 and 7). However, by extending the paths $1-5$, $6-10$ and $11-15$ to paths with length $2k$ and picking vertices incident to the leaves, we need $(k+1)$ steps in the look-ahead to detect a suboptimal choice.

The next proposition verifies that this graph and the one used in Example 3.8 are both perfect but not in the families covered by our results.

Proposition 3.11. *The graphs in Examples 3.8 and 3.10 are perfect but not generalized split, chordal, or co-chordal*

Proof. Consider the graph G in Example 3.8. We first prove the perfectness. It is clear that G is 3-colorable by coloring $\{1,4,6,8,10\}$, $\{2,5,7,9,11\}$ and $\{3\}$ differently, and the clique number of G is 3. Then, except for G, the components of any other induced subgraph of G are made of isolated vertices, paths, even holes or containing a triangle, so their chromatic numbers are $1,2,3$ respectively. That is, the clique number of any subgraph of G equals to its chromatic number, which implies the perfectness of G.

It is not hard to see that any induced subgraph of a generalized split graph is also generalized split. Let $S=\{2,3,6,7,8,9,10,11\}$ and $G|_S$. $G|_S$ is shown to be not generalized split in [16]. Thus, G is not generalized split.

Since $G|_{\{3,4,5,6\}}$ is an even hole, G is not chordal. Similarly, $\bar{G}|_{\{1,2,10,11\}}$ is an even hole, so $\bar{G}$ is not chordal. Thus, G is not chordal or co-chordal.

Consider G in Example 3.10. It is 3-colorable by coloring $\{2,4,7,9,11,13,15,17,20\}$, $\{1,3,4,19,21\}$, $\{6,8,10,12,14,16,18\}$ differently. Similar arguments as above apply; the clique number of all induced subgraphs of G equal to its chromatic number, so G is perfect.

Notice that graph in Example 3.8 is an induced subgraph of G, so G is not generalized split. Similarly, it is not chordal or co-chordal. □

4 Rounding via SDP VFA

In the previous sections, we show that Algorithm 1 with $\mathscr{V}_{\text{LP}}$ returns an MWSS for generalized split graphs, chordal graphs, and co-chordal graphs. However, constructing $\mathscr{V}_{\text{LP}}$ requires a strictly complementary solution of (LP-D), which generally takes exponential time since there is a variable for each clique in G. In this section we show that Algorithm 1 with $\mathscr{V}_{\text{SDP}}$ inherits the performance guarantees we obtain for $\mathscr{V}_{\text{LP}}$. In particular, Algorithm 1 with $\mathscr{V}_{\text{SDP}}$ generates an MWSS for generalized split graphs, chordal graphs, and co-chordal graphs.

Throughout this section, $G=(N,E)$ is a perfect graph; let (x^*,μ^*) denote a fixed pair of strictly complementary solutions of (LP-P), (LP-D), let $(\bar{x},\bar{X},\bar{q},\bar{Q})$ be a fixed tuple of optimal solutions of (SDP-P), (SDP-D) in the relative interior of the optimal face, and let t^* be the optimal value of these problems. Finally, let $\mathscr{V}_{\text{LP}}$ and $\mathscr{V}_{\text{SDP}}$ be the VFAs constructed from them.

Theorem 4.1. *Let $G=(N,E)$ be a perfect graph and let $\mathscr{V}_{LP}$ and $\mathscr{V}_{SDP}$ be as defined above. If Algorithm 1 with $\mathscr{V}_{LP}$ returns an MWSS of G, irrespective of which vertex choices are made in line 6, then Algorithm 1 with $\mathscr{V}_{SDP}$ also returns an MWSS.*

Our main result, namely Theorem 2.11, is a consequence of Theorems 2.10 and 4.1. The key idea behind the proof of Theorem 4.1 is to show that if S is on an optimal trajectory and $I\subseteq N$ is the set of remaining vertices, then for every $i\in I$,

$$\mathscr{V}_{\text{LP}}(I)-\mathscr{V}_{\text{LP}}(I\backslash(\{i\}\cup\delta_i)>w_i \implies \mathscr{V}_{\text{SDP}}(I)-\mathscr{V}_{\text{SDP}}(I\backslash(\{i\}\cup\delta_i)>w_i. \tag{5}$$

In particular, (5) shows that $\mathscr{V}_{\text{SDP}}$ discards every vertex discarded by $\mathscr{V}_{\text{LP}}$. Furthermore, by the first property of Lemma 2.4, $\mathscr{V}_{\text{SDP}}$ might discard some suboptimal vertices that $\mathscr{V}_{\text{LP}}$ does not.

Roughly, the argument to prove (5) proceeds as follows; we provide the details below. When S is on an optimal trajectory and $\mathscr{V}_{\text{LP}}(I) - \mathscr{V}_{\text{LP}}(I \setminus (\{i\} \cup \delta_i) > w_i$, there is an essential clique C of $G|_I$ in $\delta_i \cap I$; this C is a subset of an essential clique $\tilde{C}$ of G. Given the LP solution μ^*, we construct an optimal solution $(t^*, q_{\text{LP}}, Q_{\text{LP}})$ for (SDP-D), and a vector $p_{\tilde{C}} \in \mathbb{R}^n$ with $p_{\tilde{C}} \in \text{Range}(Q_{\text{LP}})$. Since $(\bar{q}, \bar{Q})$ is in the relative interior, $p_{\tilde{C}} \in \text{Range}(Q_{\text{LP}}) \subseteq \text{Range}(\bar{Q})$. By analyzing $(\bar{q}, \bar{Q})$, we show that $p_{\tilde{C}} \in \text{Range}(\bar{Q})$ implies $\mathscr{V}_{\text{SDP}}(I) - \mathscr{V}_{\text{SDP}}(I \setminus (\{i\} \cup \delta_i)) > w_i$.

4.1 Proof of Theorem 4.1

We first analyze **Phase I** of Algorithm 1 with $\mathscr{V}_{\text{SDP}}$.

Lemma 4.2. *For any $i \in V$, $\bar{x}_i > 0$ if and only if i is in an MWSS of G.*

Proof. ($\Longrightarrow$) Since G is a perfect graph, by (3) and the optimality of $\bar{x}$, there exists an MWSS of G including i. ($\Longleftarrow$) For the sake of contradiction, suppose $\bar{x}_i = 0$. By (3), there exists an optimal solution (x^*, X^*) such that $x^*_i = 1$. Then, for any $\lambda \in \mathbb{R}$, $(\bar{x}, \bar{X}) + \lambda(x^* - \bar{x}, X^* - \bar{X})$ is in the affine hull of the optimal face of (SDP-D). By $(\bar{x}, \bar{X})$ being in the relative interior of the optimal face, $(\tilde{x}, \tilde{X}) = (\bar{x}, \bar{X}) - \varepsilon(x^* - \bar{x}, X^* - \bar{X})$ is in the optimal face for $\varepsilon > 0$ small enough. But $\tilde{x}_i = \bar{x}_i - \varepsilon x^*_i < 0$, which is not feasible to (SDP-D), contradiction. □

Thus, for $\mathscr{V}_{\text{SDP}}$, Algorithm 1 discards all vertices that are not in any MWSS during **Phase I**. Hence, $\mathscr{V}_{\text{LP}}$ and $\mathscr{V}_{\text{SDP}}$ reach the same set of vertices after **Phase I**, and it remains to analyze **Phase II**. We now show how to construct an optimal solution of (SDP-D) from one of (LP-D).

Lemma 4.3. *Consider vectors $b \in \mathbb{R}^N$ and $p_C \in \mathbb{R}^N$ for $C \in \mathscr{C}(G)$, with entries*

$$b_i = \begin{cases} 0, & \text{if } i = 0 \\ \sum_{C \ni i} \mu^*_C - w_i, & \text{if } i > 0 \end{cases} \quad \text{and} \quad [p_C]_i := \begin{cases} \sqrt{\mu^*_C}, & \text{if } i = 0 \\ -\sqrt{\mu^*_C}, & \text{if } i \in C \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

Then $M := \sum_{C \in \mathscr{C}(G)} p_C p_C^\top + \text{Diag}(b)$ is optimal for (SDP-D).

Proof. First we verify feasibility. By μ^* being feasible for (LP-D), we know $\sum_{C \ni i} \mu^*_C - w_i \geq 0$, so $\text{Diag}(b)$ is PSD. Since each $p_C p_C^\top$ is PSD, M is PSD. Notice that $M_{ij} = 0$ for $ij \notin E, i \neq j$, since no clique contains both i, j and so $[p_C p_C^\top]_{ij} = 0$. Consider the blocks $M = \left(\begin{smallmatrix} t & p^\top \\ p & P \end{smallmatrix}\right)$. Since $P_{ii} = \sum_{C \ni i} \mu^*_C + \sum_{C \ni i} \mu^*_C - w_i = 2\sum_{C \ni i} \mu^*_C - w_i$ and $p_i = -\sum_{C \ni i} \mu^*_C$, then M is feasible for (SDP-D). Finally, $t = \sum_{C \in \mathscr{C}(G)} \mu^*_C = t^*$, so M is optimal. □

The next lemma establishes when $\mathscr{V}_{\text{SDP}}(I) = \mathscr{V}_{\text{SDP}}(S)$ for some $S \subseteq I$.

Lemma 4.4. *Let $\left(\begin{smallmatrix} A & C^\top \\ C & D \end{smallmatrix}\right) \succeq 0$, with $A \in \mathbb{S}^s$, $C \in \mathbb{R}^{(n-s)\times s}$ and $D \in \mathbb{S}^{n-s}$, and let $a \in \text{Range}(A) \subseteq \mathbb{R}^s$, $d \in \mathbb{R}^{n-s}$. Consider*

$$v_I := \min\left\{ t : \begin{pmatrix} t & a^\top & d^\top \\ a & A & C^\top \\ d & C & D \end{pmatrix} \succeq 0 \right\} = \begin{pmatrix} a \\ d \end{pmatrix}^\top \begin{pmatrix} A & C^\top \\ C & D \end{pmatrix}^\dagger \begin{pmatrix} a \\ d \end{pmatrix};\ v_S := \min\left\{ t : \begin{pmatrix} t & a^\top \\ a & A \end{pmatrix} \succeq 0 \right\} = a^\top A^\dagger a.$$

Let $r = \text{rank}(A)$ and assume that the upper $r \times r$ principal matrix is invertible. So we may write $A = \left(\begin{smallmatrix} \tilde{A} & \tilde{C}^\top \\ \tilde{C} & \tilde{D} \end{smallmatrix}\right)$, $\tilde{A} \in \mathbb{S}^r_{++}$, $\tilde{C} = U\tilde{A} \in \mathbb{R}^{(s-r)\times r}$, $\tilde{D} = U\tilde{A}U^\top \in \mathbb{S}^{s-r}$ for some $U \in \mathbb{R}^{(s-r)\times r}$. Using that $a \in \text{Range}(A)$, we may also write $a = \left(\begin{smallmatrix} \tilde{a} \\ \tilde{d} \end{smallmatrix}\right)$, $\tilde{a} \in \mathbb{R}^r, \tilde{d} = U\tilde{a} \in \mathbb{R}^{s-r}$, $C = (C_1\ C_2)$, $C_1 \in \mathbb{R}^{(n-s)\times r}, C_2 \in \mathbb{R}^{(n-s)\times(s-r)}$. Then, $v_I = v_S$ if and only if there exists some $V \in \mathbb{R}^{(n-r)\times r}$ such that $\left(\begin{smallmatrix} \tilde{d} \\ d \end{smallmatrix}\right) = V\tilde{a}$, $\left(\begin{smallmatrix} \tilde{C} \\ C_1 \end{smallmatrix}\right) = V\tilde{A}$, $\left(\begin{smallmatrix} \tilde{D} & C_2^\top \\ C_2 & D \end{smallmatrix}\right) \succeq V\tilde{A}V^\top$.

Proof. ($\Longleftarrow$) Assume such a V exists. The inequality $v_I \geq v_S$ always holds, so it remains to see that $v_S \geq v_I$. Let t be feasible for the SDP defining v_S, i.e., $\left(\begin{smallmatrix} t & \tilde{a}^\top \\ \tilde{a} & \tilde{A} \end{smallmatrix}\right) \succeq 0$. Since

$$\begin{pmatrix} t & a^\top & d^\top \\ a & A & C^\top \\ d & C & D \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & I \\ 0 & V \end{pmatrix} \begin{pmatrix} t & \tilde{a}^\top \\ \tilde{a} & \tilde{A} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & I & V^\top \end{pmatrix} + \begin{pmatrix} 0 & 0 & \\ 0 & 0 & \\ 0 & 0 & \begin{pmatrix} \tilde{D} & C_2^\top \\ C_2 & D \end{pmatrix} - V\tilde{A}V^\top \end{pmatrix},$$

the above matrix is PSD, so t is also feasible for the SDP defining v_I; hence, $v_S \geq v_I$.

($\Longrightarrow$) Suppose now that $v_I = v_S$. Consider first the case that $r = s$, i.e., A is positive definite. Notice that if $v_I = v_S$, then the same still holds if we replace D by $D+P$ for any PSD matrix P. Hence, we may assume that $S := D - CA^{-1}C^\top$ is also positive definite. Recall the inverse formula for a block matrix:

$$\begin{pmatrix} A & C^\top \\ C & D \end{pmatrix}^{-1} = \begin{pmatrix} A^{-1}+A^{-1}C^\top S^{-1}CA^{-1} & -A^{-1}C^\top S^{-1} \\ -S^{-1}CA^{-1} & S^{-1} \end{pmatrix}.$$

Using $v_I = \begin{pmatrix} a & d \end{pmatrix}\begin{pmatrix} A & C^\top \\ C & D \end{pmatrix}^{-1}\begin{pmatrix} a \\ d \end{pmatrix}, v_S = a^\top A^{-1}a$, we obtain

$$v_I - v_S = a^\top A^{-1}C^\top S^{-1}CA^{-1}a - 2a^\top A^{-1}C^\top S^{-1}d + d^\top S^{-1}d = \|S^{-1/2}CA^{-1}a - S^{-1/2}d\|^2.$$

Since $v_I = v_S$, then $d = CA^{-1}a$. The wanted matrix is $V = \begin{pmatrix} U \\ CA^{-1} \end{pmatrix}$.

Consider now the case that $r = \operatorname{rank}(A) < s$. Let $v_{\tilde{S}} := \min\left\{t : \begin{pmatrix} t & \tilde{a}^\top \\ \tilde{a} & \tilde{A} \end{pmatrix} \succeq 0\right\} = \tilde{a}^\top \tilde{A}^{-1}\tilde{a}$. By the first part of this proof, we get that $v_S = v_{\tilde{S}}$. Since $v_I = v_{\tilde{S}}$ and $\tilde{A} \succ 0$, we are back in the positive definite case, and we can find a matrix $\tilde{V} \in \mathbb{R}^{(n-r)\times r}$. Letting $V = \begin{pmatrix} \tilde{C} \\ C_1 \end{pmatrix}\tilde{A}^{-1} = \begin{pmatrix} U \\ \tilde{V} \end{pmatrix}$, we have $\begin{pmatrix} \tilde{d} \\ d \end{pmatrix} = V\tilde{a}$, $\begin{pmatrix} \tilde{C} \\ C_1 \end{pmatrix} = V\tilde{A}$, $\begin{pmatrix} \tilde{D} & C_2^\top \\ C_2 & D \end{pmatrix} \succeq V\tilde{A}V^\top$, as required. □

The following lemma is the key to prove Theorem 4.1.

Lemma 4.5. *Let $G = (N, E)$ and consider Algorithm 1 applied to $\mathscr{V}_{LP}$ and $\mathscr{V}_{SDP}$. Consider an arbitrary iteration of **Phase II** on an optimal trajectory, starting with the selected set of vertices S and the set of remaining vertices I. Then for $J \subsetneq I$, if $\mathscr{V}_{LP}(I) > \mathscr{V}_{LP}(J)$, we have $\mathscr{V}_{SDP}(I) > \mathscr{V}_{SDP}(J)$.*

Before proving Lemma 4.5, we first show that it implies Theorem 4.1.

Lemma 4.5 $\Longrightarrow$ Theorem 4.1. Let $S = \{v_1, \ldots, v_t\}$ be a stable set returned by Algorithm 1 with $\mathscr{V}_{\text{SDP}}$, where v_j is the j-th vertex selected in line 6. We claim that $v_1, \ldots, v_t$ are also valid choices of selected vertices when using Algorithm 1 with $\mathscr{V}_{\text{LP}}$. If we prove this claim, then S would be an MWSS by the assumption on Algorithm 1 with $\mathscr{V}_{\text{LP}}$.

Let $S_i = \{v_1, \ldots, v_{i-1}\}$ be the set of selected vertices and let $I_i^{\text{SDP}} \subseteq N \setminus S_i$ be the set of remaining vertices before v_i is selected by Algorithm 1 with $\mathscr{V}_{\text{SDP}}$. We have that $S_{i+1} = S_i \cup \{v_i\}$ and $I_{i+1}^{\text{SDP}} \subseteq I_i^{\text{SDP}} \setminus (v_i \cup \delta_{v_i})$. We will prove by induction on i that Algorithm 1 with $\mathscr{V}_{\text{LP}}$ can select the vertices in S_i for the first $i-1$ rounds, and that the set of remaining vertices I_i^{LP} is a superset of I_i^{SDP}.

For the base case, I_1^{LP}, I_1^{SDP} are the sets of remaining vertices after **Phase I** of Algorithm 1. Since $S_1 = \emptyset$ is on an optimal trajectory, the LP solution is strictly complementary, and the SDP solution is in the relative interior of the optimal face, by Section 3.1 and Lemma 4.2 we have $I_1^{\text{LP}} = I_1^{\text{SDP}}$. Thus, Algorithm 1 with $\mathscr{V}_{\text{LP}}$ can select $v_1 \in I_1^{\text{LP}}$.

Assume now that Algorithm 1 with $\mathscr{V}_{\text{LP}}$ selects the vertices in S_i, and let $I_i^{\text{LP}} \supseteq I_i^{\text{SDP}}$ be the set of remaining vertices. Since $v_i \in I_i^{\text{SDP}} \subseteq I_i^{\text{LP}}$, then we can select vertex v_i in Algorithm 1 with $\mathscr{V}_{\text{LP}}$. Notice that (5) holds, as it is a special case of Lemma 4.5. Hence, after v_i is selected, any vertex in I_i^{SDP} (a subset of I_i^{LP}) that is discarded using $\mathscr{V}_{\text{LP}}$ can also be discarded when using $\mathscr{V}_{\text{SDP}}$. It follows that $I_{i+1}^{\text{SDP}} \subseteq I_{i+1}^{\text{LP}}$. □

Proof of Lemma 4.5. For the sake of contradiction, suppose that $\mathscr{V}_{\text{SDP}}(I) = \mathscr{V}_{\text{SDP}}(J)$. The proof strategy consists of constructing two matrices M_1, M_2 with $M_1 \succeq M_2$ but such that $\mathscr{V}_{\text{LP}}(J) = \mathscr{V}_{\text{LP}}(I)$ implies $M_1 \not\succeq M_2$.

Let $J' \subseteq J$ be such that $\bar{Q}_{J'}$ is positive definite and $\operatorname{rank}(\bar{Q}_{J'}) = \operatorname{rank}(\bar{Q}_J)$. Define $\delta_I := \{i \in N \setminus I : \exists j \in I, ij \in E(G)\}$. By Lemma 4.4, there exists a matrix $V \in \mathbb{R}^{|I\setminus J'|\times|J'|}$ such that

$$M_1 := \begin{pmatrix} t^* & \bar{q}_{J'}^\top & \bar{q}_{J'}^\top V^\top & \bar{q}_{\delta_I}^\top \\ & \mathrm{R} & & \bar{Q}_{J',\delta_I} \\ V\bar{q}_{J'} & V\bar{Q}_{J'} & \bar{Q}_{I\setminus J'} & \bar{Q}_{I\setminus J',\delta_I} \\ \bar{q}_{\delta_I} & \bar{Q}_{\delta_I,J'} & \bar{Q}_{\delta_I,I\setminus J'} & \bar{Q}_{\delta_I} \end{pmatrix} \succeq M_2 := \begin{pmatrix} t^* & \bar{q}_{J'}^\top & \bar{q}_{J'}^\top V^\top & \bar{q}_{\delta_I}^\top \\ & \mathrm{R} & & \bar{Q}_{J',\delta_I} \\ & \mathrm{VR} & & \bar{Q}_{I\setminus J',\delta_I} \\ \bar{q}_{\delta_I} & \bar{Q}_{\delta_I,J'} & \bar{Q}_{\delta_I,I\setminus J'} & \bar{Q}_{\delta_I} \end{pmatrix}.$$

where $\bar{Q}_{I\setminus J'} \succeq V\bar{Q}_{J'}V^\top$, $R := \begin{pmatrix} \bar{q}_{J'} & \bar{Q}_{J'} & \bar{Q}_{J'}V^\top \end{pmatrix}$

Our goal is to show that there exists $v \in \mathbb{R}^{\{0\}\cup J'\cup(I\setminus J')\cup\delta_I}$ such that $v^\top M_1 v < v^\top M_2 v$, which would be a contradiction. We proceed to construct the vector v. By the way $\mathscr{V}_{\text{LP}}$ is constructed and $\mathscr{V}_{\text{LP}}(J) < \mathscr{V}_{\text{LP}}(I)$, we know that there is an essential clique $C' \in \mathscr{C}(G|_I)$ such that $C' \cap J = \emptyset$. And there exists a strictly essential clique $\bar{C}$ of G such that $C' \subseteq \bar{C}$. With this $\bar{C}$ and $\mu_{\bar{C}}^* > 0$, we construct a vector $p_{\bar{C}} \in \mathbb{R}^{\{0\}\cup N}$ and a matrix M as in Lemma 4.3, where M is shown to be

an optimal solution of (SDP-D) over G. By definition of M, it is clear that $p_{\bar{C}} \in \text{Range}(M)$ by [2, Theorem 1]. Let $\mathscr{H}(G)$ be the feasible set of (SDP-D) defined over G. Since $\begin{pmatrix} t^* & \bar{q}^\top \\ \bar{q} & \bar{Q} \end{pmatrix}$ is in the relative interior of the optimal face of $\mathscr{H}(G)$, or equivalently, the optimal face is the minimal face of $\mathscr{H}(G)$ containing $\begin{pmatrix} t^* & \bar{q}^\top \\ \bar{q} & \bar{Q} \end{pmatrix}$, by [11, Lemma 4] we have

$$p_{\bar{C}} \in \text{Range}(M) \subseteq \text{Range}\begin{pmatrix} t^* & \bar{q}^\top \\ \bar{q} & \bar{Q} \end{pmatrix}.$$

For the rest of the proof, let Ξ be $N \setminus (I \cup \delta_I)$, and consider the block structure

$$\begin{pmatrix} t^* & \bar{q}^\top \\ \bar{q} & \bar{Q} \end{pmatrix} = \begin{pmatrix} t^* & \bar{q}_I^\top & \bar{q}_{\delta_I}^\top & \bar{q}_\Xi^\top \\ \bar{q}_I & \bar{Q}_I & \bar{Q}_{I,\delta_I} & 0 \\ \bar{Q}_{\delta_I} & \bar{Q}_{\delta_I,I} & \bar{Q}_{\delta_I} & \bar{Q}_{\delta_I,\Xi} \\ \bar{q}_\Xi^\top & 0 & \bar{Q}_{\Xi,\delta_I} & \bar{Q}_\Xi \end{pmatrix}.$$

Let $\tilde{p}_{C'} \in \mathbb{R}^{\{0\}\cup I\cup\Xi}$ be the restriction of $p_{\bar{C}}$ to $\{0\}\cup I\cup\Xi$. Since $p_{\bar{C}} \in \text{Range}\begin{pmatrix} t^* & \bar{q}^\top \\ \bar{q} & \bar{Q} \end{pmatrix}$, then $\tilde{p}_C \in \text{Range}\begin{pmatrix} t^* & \bar{q}_I^\top & \bar{q}_\Xi^\top \\ \bar{q}_I & \bar{Q}_I & 0 \\ \bar{q}_\Xi^\top & 0 & \bar{Q}_\Xi \end{pmatrix}$.

Hence, there exists $\tilde{y} \in \mathbb{R}^{\{0\}\cup I\cup\Xi}$ such that $\tilde{p}_C = \begin{pmatrix} t^* & \bar{q}_I^\top & \bar{q}_\Xi^\top \\ \bar{q}_I & \bar{Q}_I & 0 \\ \bar{q}_\Xi^\top & 0 & \bar{Q}_\Xi \end{pmatrix}\tilde{y}$. Consider the vectors

$$\ell := \begin{pmatrix} 1 & \bar{x}_{J'}^\top & \bar{x}_{I\setminus J'}^\top & \bar{x}_{\delta_I}^\top \end{pmatrix}^\top, \quad \bar{y} := \begin{pmatrix} \tilde{y}_{\{0\}}^\top & \tilde{y}_{J'}^\top & \tilde{y}_{I\setminus J'}^\top & 0 \end{pmatrix}, \quad \nu_\gamma = \bar{y} + \gamma\ell \in \mathbb{R}^{\{0\}\cup J'\cup(I\setminus J')\cup\delta_I}$$

It remains to show that $\nu_\gamma^\top M_1 \nu_\gamma < \nu_\gamma^\top M_2 \nu_\gamma$ for a suitable choice of γ.

Let us first evaluate $M_1\bar{y}$ and $M_2\bar{y}$. Consider the equation $\begin{pmatrix} t^* & \bar{q}_I^\top & \bar{q}_\Xi^\top \\ \bar{q}_I & \bar{Q}_I & 0 \\ \bar{q}_\Xi^\top & 0 & \bar{Q}_\Xi \end{pmatrix}\tilde{y} = \tilde{p}_C$. By replacing I by $J' \cup (I \setminus J')$ and adding rows and columns corresponding to δ_I, we can rewrite this equation as

$$\begin{pmatrix} t^* & \bar{q}_{J'}^\top & \bar{q}_{J'}^\top V^\top & \bar{q}_{\delta_I}^\top & \bar{q}_\Xi^\top \\ \bar{q}_{J'} & \bar{Q}_{J'} & \bar{Q}_{J'}V^\top & \bar{Q}_{J',\delta_I} & 0 \\ V\bar{q}_{J'} & V\bar{Q}_{J'} & \bar{Q}_{I\setminus J'} & \bar{Q}_{I\setminus J',\delta_I} & 0 \\ \bar{q}_{\delta_I} & \bar{Q}_{\delta_I,J'} & \bar{Q}_{\delta_I,I\setminus J'} & \bar{Q}_{\delta_I} & \bar{Q}_{\delta_I,\Xi} \\ \bar{q}_\Xi & 0 & 0 & \bar{Q}_{\Xi,\delta_I} & \bar{Q}_\Xi \end{pmatrix}\begin{pmatrix} \tilde{y}_{\{0\}} \\ \tilde{y}_{J'} \\ \tilde{y}_{I\setminus J'} \\ 0 \\ \tilde{y}_\Xi \end{pmatrix} = (\underbrace{\sqrt{\mu_{\bar{C}}^*}}_{\{0\}}\ \underbrace{0}_{J'}\ \underbrace{-\sqrt{\mu_{\bar{C}}^*}\cdots-\sqrt{\mu_{\bar{C}}^*}}_{C'}\ \underbrace{0}_{I\setminus J'\setminus C'}\ \underbrace{*\cdots*}_{N\setminus I})^\top,$$

where $*\cdots*$ represent irrelevant entries. Since $\bar{y} = \begin{pmatrix} \tilde{y}_{\{0\}} & \tilde{y}_{J'} & \tilde{y}_{I\setminus J'} & 0 \end{pmatrix}$, then

$$M_1\bar{y} = \begin{pmatrix} t^* & \bar{q}_{J'}^\top & \bar{q}_{J'}^\top V^\top & \bar{q}_{\delta_I}^\top \\ \bar{q}_{J'} & \bar{Q}_{J'} & \bar{Q}_{J'}V^\top & \bar{Q}_{J',\delta_I} \\ V\bar{q}_{J'} & V\bar{Q}_{J'} & \bar{Q}_{I\setminus J'} & \bar{Q}_{I\setminus J',\delta_I} \\ \bar{q}_{\delta_I} & \bar{Q}_{\delta_I,J'} & \bar{Q}_{\delta_I,I\setminus J'} & \bar{Q}_{\delta_I} \end{pmatrix}\bar{y} = (\underbrace{\alpha}_{\{0\}}\ \underbrace{0}_{J'}\ \underbrace{-\sqrt{\mu_{\bar{C}}^*}\cdots-\sqrt{\mu_{\bar{C}}^*}}_{C'}\ \underbrace{0}_{I\setminus J'\setminus C'}\ \underbrace{*\cdots*}_{\delta_I})^\top,$$

for a constant α. While for M_2, by $[M_1]_{J',:}\,\bar{y} = [R \quad \bar{Q}_{J',\delta_I}]\bar{y} = 0$, the row dependence of M_2 and $\bar{y}_{\delta_I} = 0$, we have

$$M_2\bar{y} = (\alpha \quad 0 \quad 0 \quad 0 \quad *\cdots*)^\top.$$

Since we are following the optimal trajectory, all vertices in δ_I are discarded, and $I \setminus J$ contains an essential clique.

We now evaluate $M_1\ell$ and $M_2\ell$. Since we are following an optimal trajectory and reach I, we know there exists an MWSS containing no vertices in δ_I. Thus, by (3), there exists an optimal solution $(\tilde{x}, \tilde{X})$ of (SDP-P) such that $\tilde{x}_{\delta_I} = 0$. By complementary slackness, $\begin{pmatrix} t^* & \bar{q}^\top \\ \bar{q} & \bar{Q} \end{pmatrix}\begin{pmatrix} 1 \\ \tilde{x} \end{pmatrix} = 0$. Also, since $\bar{C}$ is strictly essential over G, $\bar{C} \setminus C' \subseteq \delta_I$ and $\tilde{x}_{\delta_I} = 0$, $\tilde{x}_j > 0$ for some $j \in C' \subseteq I \setminus J$. Since $\ell := \begin{pmatrix} 1 & \bar{x}_{J'}^\top & \bar{x}_{I\setminus J'}^\top & \bar{x}_{\delta_I}^\top \end{pmatrix}^\top$, then $\ell_j > 0$ and $\ell_{\delta_I} = 0$. Since $\bar{Q}_{I,\Xi} = 0$, then

$$\begin{pmatrix} t^* & \bar{q}^\top \\ \bar{q} & \bar{Q} \end{pmatrix}\begin{pmatrix} 1 \\ \tilde{x} \end{pmatrix} = 0 \quad \Longrightarrow \quad \begin{pmatrix} R & \bar{Q}_{J',\delta_I} \\ V\bar{q}_{J'} & V\bar{Q}_{J'} & \bar{Q}_{I\setminus J'} & \bar{Q}_{I\setminus J',\delta_I} \end{pmatrix}\ell = 0,$$

and by the row dependence of M_2, we have $M_1\ell = (\lambda \ 0 \ 0 \ 0 \ *\cdots*)^\top = M_2\ell$ for a constant λ.

We proceed to evaluate $\nu_\gamma^\top M_1 \nu_\gamma$ and $\nu_\gamma^\top M_2 \nu_\gamma$. Since $\ell_{\delta_I} = \tilde{x}_{\delta_I} = 0$,

$$\nu_\gamma^\top M_1 \nu_\gamma = \bar{y}^\top M_1\bar{y} + 2\gamma\ell^\top M_1\bar{y} + \gamma^2\ell^\top M_1\ell = \bar{y}^\top M_1\bar{y} + 2\gamma\alpha + 2\gamma\sum_{i\in C'}\left(-\sqrt{\mu_{\bar{C}}^*}\right)\ell_i + 0 + \lambda\gamma^2$$

$$\nu_\gamma^\top M_2 \nu_\gamma = \bar{y}^\top M_2\bar{y} + 2\gamma\ell^\top M_2\bar{y} + \gamma^2\ell^\top M_2\ell = \bar{y}^\top M_2\bar{y} + 2\gamma\alpha + \lambda\gamma^2.$$

Since $-\sqrt{\mu^*_{\bar{C}}} < 0$, $\ell \geq 0$ and $\ell_j = \tilde{x}_j > 0$, $j \in C' \subseteq I \setminus J$, then $v_\gamma^\top M_1 v < v_\gamma^\top M_2 v_\gamma$ for large enough γ. This contradicts the fact that $M_1 \succeq M_2$. □

We emphasize that even though the analysis of Algorithm 1 with $\mathscr{V}_{\text{SDP}}$ relies on $\mathscr{V}_{\text{LP}}$, there is no need to compute $\mathscr{V}_{\text{LP}}$ in practice.

4.2 Computing all Maximum Weighted Stable Sets in Polynomial Time

We now show how to modify Algorithm 1 to efficiently obtain all MWSS for unipolar, chordal and co-chordal graphs. In addition to the number of vertices n, we measure the algorithm's efficiency with the number of solutions it generates; Algorithm 2 describes the approach.

Algorithm 2 Retrieving all MWSS from a tight VFA with one-step look ahead

Input: $G = (N,E)$, a weight function w, optimal $x^* \in \text{STAB}(G)$, and a tight VFA $\mathscr{V}$.
Phase I:
$\mathscr{S} \leftarrow \emptyset$ ▷ List of stable sets, initially empty.
$I \leftarrow \{i \in N : x^*_i > 0\}$ ▷ Discard vertices not in any optimal set.
Index elements in I from 1 to $|I|$.
$T \leftarrow [(I, \emptyset)]$ ▷ List of unvisited tree nodes.
Phase II:
while $T \neq \emptyset$ **do**
  $(J,S) \leftarrow \text{pop}(T)$ ▷ Get the last element in T and discard it.
  if $J = \emptyset$ **then**
    $\mathscr{S} \leftarrow \mathscr{S} \cup \{S\}$
  else
    $v \leftarrow u \in J$ with the smallest index
    $I' \leftarrow J \setminus \{v\}$ ▷ Test deleting v
    while $\exists i \in I'$ with $\mathscr{V}(I') - \mathscr{V}(I' \setminus (\{i\} \cup \delta_i)) > w_i$ **do**
      $I' \leftarrow I' \setminus \{i\}$
    if $\mathscr{V}(J) = \mathscr{V}(I')$ **then**
      $T \leftarrow \text{append}(T, (I', S))$ ▷ Append the element to the end of T
    $I' \leftarrow J$ ▷ Test adding v to S
    $I' \leftarrow I' \setminus (\{v\} \cup \delta_v)$
    while $\exists i \in I'$ with $\mathscr{V}(I') - \mathscr{V}(I' \setminus (\{i\} \cup \delta_i)) > w_i$ **do**
      $I' \leftarrow I' \setminus \{i\}$
    if $\mathscr{V}(J) = \mathscr{V}(I') + w_v$ **then**
      $T \leftarrow \text{append}(T, (I', S \cup \{v\}))$. ▷ Append the element to the end of T
Output: Return $\mathscr{S}$, a list of stable sets of G.

To establish the algorithm's correctness, we first show that, for unipolar, chordal and co-chordal graphs, if a vertex is essential, deleting it decreases the VFA.

Lemma 4.6. *Let $G = (N,E)$ be either unipolar, chordal or co-chordal and consider Algorithm 1 applied to $\mathscr{V}_{LP}$. Consider an iteration of **Phase II** starting with set of vertices I; if v is in every MWSS of $G|_I$, then $\mathscr{V}_{LP}(I) > \mathscr{V}_{LP}(I \setminus \{v\})$.*

Proof. For contradiction, suppose that $\mathscr{V}_{\text{LP}}(I) = \mathscr{V}_{\text{LP}}(I \setminus \{v\})$ with the set of selected vertices S. By the proof of Theorem 2.10 (a), for each later iteration starting with the set of vertices J, there exists a vertex $u \in J$ such that $\mathscr{V}_{\text{LP}}(J) = \mathscr{V}_{\text{LP}}(J \setminus (\{u\} \cup \delta_u)$. Thus, by $\mathscr{V}_{\text{LP}}$ being tight, the returned stable set $\hat{S}$ is an MWSS of G. However, since $v \notin \hat{S}$ and every MWSS of $G|_I$ contains v, we know $\hat{S} \setminus S$ is not an MWSS of $G|_I$, so $\hat{S}$ is not an MWSS of G, a contradiction. □

Corollary 4.7. *Let $G = (N,E)$ be either unipolar, chordal or co-chordal and consider Algorithm 1 applied to $\mathscr{V}_{SDP}$. Consider an iteration of **Phase II** starting with set of vertices I; if v is in every MWSS of $G|_I$, then $\mathscr{V}_{SDP}(I) > \mathscr{V}_{SDP}(I \setminus \{v\})$.*

Proof. For contradiction, suppose that $\mathscr{V}_{\text{SDP}}(I) = \mathscr{V}_{\text{SDP}}(I \setminus \{v\})$. Let $\hat{S}$ be a stable set returned by Algorithm 1. By Lemma 4.5, $\mathscr{V}_{\text{LP}}(I) = \mathscr{V}_{\text{LP}}(I \setminus \{v\})$. Thus by the proof of Lemma 4.5 implying Theorem 4.1, we know $\hat{S}$ can be returned by Algorithm 1 with $\mathscr{V}_{\text{LP}}$, so it is an MWSS of G. Using the same argument in the proof of Lemma 4.6, we see that $\hat{S}$ is not an MWSS of G, a contradiction. □

Thus, if one is on an optimal trajectory with $\mathscr{V}_{\text{SDP}}(I) = \mathscr{V}_{\text{SDP}}(I \setminus \{v\})$, then one stays on an optimal trajectory after deleting v. Now we show the main result of this subsection.

Theorem 4.8. *For unipolar, chordal and co-chordal graphs, if there are ℓ different MWSS, Algorithm 2 with $\mathscr{V}_{SDP}$ computes all ℓ MWSS in $\mathscr{O}(n^2\ell)$ VFA evaluations.*

Proof. Let $G = (N, E)$ be unipolar, chordal or co-chordal, and let $S_1, \ldots, S_\ell$ be the collection of all MWSS. By Lemma 4.2, $I := \bigcup_{i=1}^{\ell} S_i$; without loss of generality, assume $|I| = n$ and $I = \{1, \ldots, n\}$.

Consider a binary tree T where each node (J, S) indicates that J is the vertex set of the remaining graph and $S \subseteq I \setminus J$ is the selected partial stable set. For the node (J, S), let v be the vertex in J with the smallest index. Algorithm 2 creates children for this node in the following ways.

1. Consider deleting v. After tentatively deleting v, if $\mathscr{V}_{\text{SDP}}$ does not decrease in value, create a child node (I', S), where I' is the remaining vertex set after deleting v and deleting all bad vertices from J. If $\mathscr{V}_{\text{SDP}}$ decreases in value, by Corollary 4.7, v is an element of every MWSS of the remaining set; do not create a child node for deleting v.
2. Consider adding v to S and applying a look-ahead. After tentatively choosing v, if v is a good choice, create a child node $(I', S \cup \{v\})$, where I' is the remaining vertex set after choosing v, deleting its neighbors, and deleting all bad vertices from J. If v is a bad choice, do not create a child node for adding v.

Thus, at most two children are created, one for selecting v and one for deleting v. Since Algorithm 2 considers and discards the lowest-indexed remaining vertex in each iteration, it does not create duplicate nodes with the same pair (J, S). Each node is considered exactly once, and leaves are nodes of the form $(\emptyset, S)$.

Claim. *Every MWSS is represented by a unique leaf in the binary tree.*

Proof. Let $\hat{S} = \{v_1, v_2, v_3, \ldots, v_k\}$ be an arbitrary MWSS, where $v_1 < v_2 < v_3 < \cdots < v_k$. By Theorem 2.10 (a) and Lemma 2.4, $\hat{S}$ can be possibly returned by Algorithm 1 and hence by Algorithm 2. And since $\hat{S}$ is an MWSS, J is empty after v_k is selected, so it is represented uniquely by a leaf. In particular, it is represented uniquely by the path from the root to the leaf following the algorithm's order of selecting and deleting vertices according to the indexing. □

Claim. *Every leaf of the binary tree represents an MWSS of the graph.*

Proof. By Theorem 2.10 (a), Theorem 4.1, Lemma 4.6 and Corollary 4.7, for all possible actions (children) at each node, Algorithm 2 stays on an optimal trajectory. Thus, when the remaining vertex set J is not empty, there exists at least one optimal choice in J to select. If the smallest-indexed vertex $v \in J$ is an optimal choice, then a child node for "selecting v" is in the tree; if it is not in every MWSS of G', then a child node for "deleting v" is also created. Thus, for every leaf in the binary tree, J is empty. The path from the root to that leaf represents a unique stable set for G made of all nodes selecting vertices and it can be returned by Algorithm 1 and Algorithm 2. By Theorem 2.10 (a) and Theorem 4.1, it is an MWSS. □

By the two claims above, we can compute all MWSS by running Algorithm 2. Since each node of the binary tree T is visited exactly once, the number of iterations is the number of nodes. The depth of T is at most n and it has ℓ leaves, so the number of nodes is $\mathscr{O}(n\ell)$. Each node involves $\mathscr{O}(n)$ VFA evaluations, resulting in $\mathscr{O}(n^2\ell)$ total VFA evaluations for Algorithm 2. □

Since $\mathscr{V}_{\text{SDP}}$ can be computed in polynomial time given the SDP solution, Theorem 4.8 shows that all MWSS can be returned in time that is polynomial time in the number of vertices and the number of different MWSS.

5 Computational Experiments

In this section, we discuss computational experiments on a variant of Algorithm 1 applied to arbitrary graphs; we detail this version in Algorithm 3. This algorithm variant does not assume that the VFA is tight, so it can be applied with imperfect graphs. In particular, it does not discard vertices j for which $\mathscr{V}(I) > \mathscr{V}(I \setminus (\{j\} \cup \delta_j)) + w_j$, but instead selects a vertex by maximizing $\mathscr{V}(I \setminus (\{j\} \cup \delta_j)) + w_j$; this is how VFAs are typically used to generate heuristic solutions in approximate DP. To reduce the number of VFA evaluations, Algorithm 3 selects isolated vertices or optimal leaves of $G|_I$ whenever possible; the latter are leaves whose weight matches or exceeds that of their only neighbor.

Algorithm 3 Retrieving a stable set from an arbitrary VFA

Input: $G = (N, E)$, a weight function w and a VFA $\mathscr{V}$.
$S \leftarrow \emptyset$ and $I \leftarrow N$ ▷ Start with an empty stable set.
while $I \neq \emptyset$ **do**
$i \leftarrow \begin{cases} \text{an isolated vertex or optimal leaf of } G|_I & \textbf{if one exists} \\ \arg\max_{j \in I}(\mathscr{V}(I \setminus (\{j\} \cup \delta_j)) + w_j) & \textbf{otherwise} \end{cases}$
$S \leftarrow S \cup \{i\}$ ▷ Select i to join the stable set.
$I \leftarrow I \setminus (\{i\} \cup \delta_i)$ ▷ Discard i and δ_i from remaining vertices.
Output: Return S, a stable set of G.

We performed the experiments on a 2021 MacBook Pro with an 8-core Apple M1 Pro CPU and 16 GB of memory. We solve the primal-dual pair (SDP-P), (SDP-D) using either the commercial solver COPT [26] for dense graphs or the SDP solver HALLaR [43] for sparse graphs; we implement Algorithm 3 in Julia. We solve all SDPs within a relative precision of $\varepsilon_{\text{SDP}} = 10^{-5}$. We evaluate $\mathscr{V}_{\text{SDP}}$ using the quadratic minimization characterization from Lemma 2.7, $\mathscr{V}_{\text{SDP}}(S) = -\min_{y \in \mathbb{R}^S}(y^\top Q_S y - 2q_S^\top y)$, including a regularization term $\lambda \|y\|^2$ with $\lambda = 10^{-4}$. We solve the VFA quadratic minimization to a precision of $\varepsilon_{\text{VFA}} = 10^{-6}$ with an iterative method, warm-started with the previous solution; this characterization allows us to trade precision for efficiency. Note that it is possible to get better results by tuning the precision parameters to each individual instance; however, we present results using uniform parameters for consistency.

In our first set of experiments, we test Algorithm 3 on special classes of perfect graphs. More precisely, we generate chordal and co-chordal graphs using the random generators with algorithms `"growing"`, `"connecting"` and `"pruned"` from SageMath [53] with default parameters. We also generate uniformly random generalized split graphs using the algorithm by McDiarmid and Yolov [42]. In total, we tested 300 chordal and co-chordal graphs (100 generated by each algorithm) and 100 generalized split graphs, using the cardinality objective, $w_i = 1$ for every $i \in N$. For all instances, we obtain an optimal solution, which agrees with our theoretical results.

In the second set of experiments, we test Algorithm 3 on graphs that are not necessarily perfect, including graphs from DIMACS2 [33], GSet [56] DIMACS10 [7], and SNAP [36]; for DIMACS2, we use graph complements, as the instances were designed for the maximum clique problem. As before, we use the cardinality objective. We compare Algorithm 3 against the Benson-Ye (BY) rounding method [12] and the Walteros-Buchanan (WB) fixed-parameter tractable algorithm [54]. We run BY on the same graphs as Algorithm 3, and WB on the complements, as it is designed for the maximum clique problem. BY is randomized and inexpensive, so we run it $|N|$ times as suggested in [12] and report its best and average performance. For all three methods, we first apply a pre-processing step so that the graph is connected and has no leaves: if it is disconnected, we consider each component separately, and if there is a leaf, we select it. We implement BY and use the WB code provided in [54]. The latter is an exact algorithm taking exponential time in the worst case, but it performs well in practice in many large instances. However, some instances are challenging for WB; for more details, we refer the reader to [54]. We only include results for instances that are solved in 30 minutes and leave the solution time blank otherwise.

Tables 1 and 2 respectively summarize the results for DIMACS2 (78 graphs) and GSet (71 graphs). Table 3 summarizes the results for the larger instances from DIMACS10 (19 graphs) and SNAP (7 graphs). Overall, Algorithm 3 significantly outperforms the BY average in every single instance, and also beats the BY best solution (often significantly) except in instances `san200-0-7-2` and `san200-0-9-3` from DIMACS2. For these two instances, we can obtain an optimal solution by tuning the parameters $\lambda, \varepsilon_{\text{VFA}}$, but we do not include those results for consistency. WB returns the optimal value whenever it terminates within the time limit, but does not terminate for many of our instances. We highlight in bold the instances where Algorithm 3 matches α or the best known bound per [51, 54].

Our algorithm's performance is noteworthy, particularly in the large instances. The stable set generated by the algorithm frequently attains the stability number α when this quantity is known; in instances where it falls short or where α is unknown, our method still significantly outperforms the BY method. Finally, note that the computational effort to run Algorithm 3 is typically much smaller than the cost of solving the SDP, so we expect it to perform much more efficiently than [30] in practice.

To highlight the versatility of our algorithm, we conducted a third set of experiments on instances with a weighted objective. We reuse the 300 chordal and co-chordal graphs and the 100 generalized split graphs generated in the first set of experiments. For an instance with name G and n vertices, we initialized a random number generator `MersenneTwister` in Julia with the per-instance seed `hash((20260226, G, n))` and sampled n weights uniformly from $\{1, \ldots, 10\}$. For all these instances, Algorithm 3 obtains a maximum weighted stable set, which agrees with our theoretical results.

We also conducted an analogous set of experiments with weighted-objective instances based on the DIMACS2 dataset. The computational setup is identical to the earlier experiments with the following exceptions. We solved instance `MANN-a45` using COSMO [22] because COPT encountered numerical issues and reported an erroneous bound; similarly, we solved instances `c2000.5, c2000.9, c4000.5, keller6` with COSMO, to within a relative precision of $\varepsilon_{\text{SDP}} = 10^{-4}$, because the weighted objective significantly increased computation times in these instances. The results are summarized in Table 4. As before, these graphs are not necessarily perfect, so the Lovász theta function bound $\vartheta(G,w)$ is only an upper bound benchmark. Since the exact weighted stability numbers for these instances are unknown and the WB algorithm is limited to the unweighted variant, we omit the corresponding baseline columns. We highlight in bold the instances where Algorithm 3 matches the upper bound ϑ. In all reported instances where $\vartheta(G,w)$ is integral, Algorithm 3 matches $\vartheta(G,w)$ and is therefore optimal.

6 Conclusions and Future Directions

We provide a novel rounding scheme for the Lovász theta function to solve the MWSS problem. Algorithm 1 relies on a VFA constructed from the optimal solution of the SDP. Theorem 2.11 guarantees that Algorithm 1 paired with VFA $\mathscr{V}_{\text{SDP}}$ solves the MWSS problem for several important subclasses of perfect graphs, which asymptotically cover almost all perfect graphs; the algorithm requires only the initial SDP solution. The algorithm can also be modified to efficiently output all MWSS for unipolar, chordal and co-chordal graphs. To the best of our knowledge, this is the only known rounding strategy for the Lovász theta function that solves the MWSS problem for large sub-classes of perfect graphs.

Our computational experiments show that Algorithm 3, a simplified variant of Algorithm 1, works well in practice. Algorithm 3 recovers an MWSS for all instances of perfect graphs we tested. The algorithm performs very well even for imperfect graphs, matching the best known bound in many of the instances we tested. We believe that Algorithm 3 should perform well on graphs for which the theta function $\vartheta(G)$ is close to the stability number $\alpha(N)$. Importantly, the computational cost of our rounding procedure is much lower than the cost of solving the SDP with a state-of-the-art method.

None of our computational experiments use look-ahead. We conjecture that the requirement of using look-ahead is only an artifact of our proof technique, and that Algorithm 1 without look-ahead still returns an MWSS for generalized split graphs, chordal, and co-chordal graphs.

Our analysis of $\mathscr{V}_{\text{SDP}}$ uses the LP-based VFA $\mathscr{V}_{\text{LP}}$. The need for $\mathscr{V}_{\text{LP}}$ comes from its combinatorial interpretation, and we do not have a similar interpretation for $\mathscr{V}_{\text{SDP}}$. An interesting future direction is finding a combinatorial interpretation of $\mathscr{V}_{\text{SDP}}$, or perhaps constructing a different VFA that optimizes the stable set problem for a larger family of perfect graphs.

Further research includes applying similar techniques to related problems, such as the dynamic stable set problem proposed in [44].

Declarations

The authors' work was partially funded by the U.S. Office of Naval Research, N00014-23-1-2631.

The authors have no other competing interests to declare that are relevant to the content of this article.

Graph					Solution value				Time (s)			
Name	$\|N\|$	$\|E\|$	ϑ	α	Alg.3	BY_{avg}	BY_{best}	WB	SDP	Alg.3	BY_{total}	WB
MANN-a27	378	702	132.77	126	126	113.2	124	126	7.92	1.02	0.02	0.16
MANN-a45	1035	1980	356.05	345	343	307.06	339	345	38.89	10.76	0.14	129.97
MANN-a81	3321	6480	1126.64	1100	1098	985.88	1089	1100	543.92	151.41	1.91	55958.3
MANN-a9	45	72	17.48	16	16	14.64	16	16	0.2	1.99	0.06	<0.01
brock200-1	200	5066	27.46	21	19	7.11	13	21	3.4	0.85	0.05	12.25
brock200-2	200	10024	14.23	12	10	2.66	6	12	18.87	0.57	0.08	0.19
brock200-3	200	7852	18.82	15	13	4.22	8	15	9.59	0.65	0.06	0.99
brock200-4	200	6811	21.29	17	14	4.73	9	17	6.93	0.74	0.05	3.22
brock400-1	400	20077	39.7	27	22	6.4	12		142.46	2.98	0.3	
brock400-2	400	20014	39.56	29	21	6.46	12		150.39	3.04	0.32	
brock400-3	400	20119	39.48	31	24	6.57	12		142.73	2.66	0.31	
brock400-4	400	20035	39.6	33	22	6.57	12		143.57	3.73	0.32	
brock800-1	800	112095	42.22	23	20	3.78	9		183.06	10.32	3.34	
brock800-2	800	111434	42.47	24	18	3.78	9		177.83	10.85	3.31	
brock800-3	800	112267	42.24	25	19	3.66	9		180.22	10.51	3.37	
brock800-4	800	111957	42.35	26	20	3.58	9		177.56	11.53	3.34	
c-fat200-1	200	18366	12	12	12	3.46	12	12	44.17	0.81	0.16	<0.01
c-fat200-2	200	16665	24	24	24	23.91	24	24	0.45	0.27	0.16	<0.01
c-fat200-5	200	11427	60.35	58	58	31.38	58	58	1.14	0.04	0.1	<0.01
c-fat500-10	500	78123	126	126	126	114.32	126	126	12.40	1.1	1.66	0.02
c-fat500-1	500	120291	14	14	14	3.18	14	14	4.17	1.83	2.12	<0.01
c-fat500-2	500	115611	26	26	26	5.86	26	26	18.12	1.27	2.03	<0.01
c-fat500-5	500	101559	64	64	64	59.12	64	64	201.85	0.84	1.99	<0.01
c1000.9	1000	49421	123.49	≥ 68	62	18.42	33		675.64	32.1	1.88	
c125.9	125	787	37.81	34	34	21.76	32		3.24	0.28	<0.01	0.45
c2000.5	2000	999164	44.87	16	14	1.84	7		1556.79	117.96	82.91	
c2000.9	2000	199468	178.93	≥ 80	68	16.41	31		35044.21	272.68	19.54	
c250.9	250	3141	56.24	44	40	21.36	31		6.53	0.75	0.03	
c4000.5	4000	3997732	64.57	18	15	1.68	6		518753.57	4840.99	680.11	
c500.9	500	12418	84.2	≥ 57	52	20.51	35		49.55	4.78	0.25	
dsjc1000_5	1000	249674	31.89	15	12	2.03	6	15	13610.94	75.32	9.63	3045.79
dsjc500_5	500	62126	22.74	13	11	2.31	6	13	890/11	9.96	1.11	179.96
gen200_p0.9_44	200	1990	44.01	44	38	22.66	38		5.10	0.98	0.02	361.19
gen200_p0.9_55	200	1990	55	55	55	54.7	55		1.81	0.24	0.02	172.09
gen400_p0.9_55	400	7980	55	55	45	20.88	33		72.46	1.55	0.14	
gen400_p0.9_65	400	7980	65.02	65	44	21.18	34		28.22	3.01	0.16	
gen400_p0.9_75	400	7980	75	75	75	74.54	75		44.17	1.66	0.14	
hamming10-2	1024	5120	512	512	512	512	512		0.1	4.07	0.34	8.62
hamming10-4	1024	89600	51.2	≥ 40	35	3.56	18		104.78	18.7	3.75	
hamming6-2	64	192	32	32	32	32	32	32	0.11	0.01	<0.01	<0.01
hamming6-4	64	1312	5.33	4	4	1.53	3	4	0.07	0.01	<0.01	<0.01
hamming8-2	256	1024	128	128	128	128	128	128	0.05	0.21	0.02	0.05
hamming8-4	256	11776	16	16	16	2.25	9	16	34.76	0.46	0.12	6.83
johnson16-2-4	120	1680	8	8	8	2.75	6	8	0.33	0.01	0.01	3.33
johnson32-2-4	496	14880	16	16	16	3.87	12		45.22	0.06	0.32	
johnson8-2-4	28	168	4	4	4	2.11	4	4	0.02	2.22	0.07	0.01
johnson8-4-4	70	560	14	14	14	5.74	14	14	0.06	0.09	<0.01	<0.01
keller4	171	5100	14.01	11	11	2.95	7	11	2.77	1.2	0.04	0.82
keller5	776	74710	31	27	21	3.82	12		79.41	4.34	2.26	
keller6	3361	1026582	66.89	59	47	9.49	39		9662.47	1275.15	147.79	
p-hat1000-1	1000	377247	17.61	10	9	1.68	6	10	241.39	26.67	14.36	54.62
p-hat1000-2	1000	254701	55.61		44	19.14	38		10622.85	49.4	10.32	
p-hat1000-3	1000	127754	84.8		63	24.8	50		6577.31	27.14	5.03	
p-hat1500-1	1500	839327	22.01	12	10	1.61	5	12	663.11	95.44	50.08	1122.13
p-hat1500-2	1500	555290	77.56	65	63	26.04	53		66738.36	171.87	35.82	
p-hat1500-3	1500	277006	115.43	94	91	34.58	74		68850.02	623.32	32.34	
p-hat300-3	300	11460	41.17	36	34	16.45	29		38.51	7.22	0.73	167.74
p-hat500-1	500	93181	13.07	9	8	1.75	6		32.59	30.22	1.79	0.54
p-hat500-2	500	61804	38.97	36	34	17.72	34		1492.97	53.84	1.33	208.03
p-hat500-3	500	30950	58.57	50	48	22.28	42		1100.74	35.55	0.66	
p-hat700-1	700	183651	15.12	11	8	1.67	7		95.68	67.09	4.94	8.09
p-hat700-2	700	122922	49.02	44	43	21.45	39		4091.29	155.49	4.69	
p-hat700-3	700	61640	72.7	62	62	26.98	54		4256.37	2.84	2.10	
san1000	1000	249000	15	15	10	1.67	8		75.06	22.29	16.00	
san200-0-7-1	200	5970	30	30	30	30	30		5.12	0.95	0.06	674.78
san200-0-7-2	200	5970	18		14	17.95	18		7.39	2.14	0.05	
san200-0-9-1	200	1990	70	70	70	70	70		0.78	1.34	0.05	3.82
san200-0-9-2	200	1990	60	60	60	60	60		0.68	0.96	0.04	130.23
san200-0-9-3	200	1990	44		37	43.88	44		1.07	2.04	0.02	1351.19
san400-0-5-1	400	39900	13	13	13	12.92	13		24.38	20.09	0.63	
san400-0-7-1	400	23940	40	40	40	40	40		219.62	9.33	0.54	
san400-0-7-2	400	23940	30	30	30	30	30		245.79	18.95	0.48	
san400-0-7-3	400	23940	22		16	4.15	9		275.76	0.14	0.36	
san400-0-9-1	400	7980	100	100	100	100	100		12.80	9.56	0.22	
sanr200-0-7	200	6032	23.84	18	16	5.76	12		5.76	1.99	0.10	4.89
sanr200-0-9	200	2037	49.3	42	41	21.92	33		0.75	0.36	0.02	966.90
sanr400-0-5	400	39816	20.32	13	12	2.49	7		31.23	11.08	0.63	12.54
sanr400-0-7	400	23931	34.28	21	19	5.27	10		247.17	20.65	0.66	

Table 1: Comparison of Algorithm 3 with Benson-Ye (BY) and Walteros-Buchanan (WB) on graphs from the DIMACS2 dataset [33]. All instances are complemented first; $|E|$ is the number of edges after taking the complement. Algorithm 3 attains optimality for the bolded instances. We run the randomized BY method $|N|$ times and report the average and best value obtained.

References

[1] Abrishami, T., Chudnovsky, M., Pilipczuk, M., Rzażewski, P., Seymour, P.: Induced subgraphs of bounded treewidth and the container method. SIAM Journal on Computing **53**(3), 624–647 (2024)

[2] Albert, A.: Conditions for positive and nonnegative definiteness in terms of pseudoinverses. SIAM Journal on Applied Mathematics **17**(2), 434–440 (1969)

[3] Alizadeh, F.: A sublinear-time randomized parallel algorithm for the maximum clique problem in perfect graphs. In: Proceedings of the Second Annual ACM-SIAM Symposium on Discrete Algorithms. p. 188–194. SODA '91, Society for Industrial and Applied Mathematics, USA (1991)

[4] Alizadeh, F.: Interior point methods in semidefinite programming with applications to combinatorial optimization. SIAM Journal on Optimization **5**(1), 13–51 (1995)

[5] Babel, L.: Finding maximum cliques in arbitrary and in special graphs. Computing **46**(4), 321–341 (1991)

[6] Babel, L., Tinhofer, G.: A branch and bound algorithm for the maximum clique problem. Zeitschrift für Operations Research **34**(3), 207–217 (1990)

[7] Bader, D.A., Meyerhenke, H., Sanders, P., Wagner, D., editors: Graph Partitioning and Graph Clustering. 10th DIMACS Implementation Challenge Workshop, Contemporary Mathematics, vol. 588. American Mathematical Society (2013)

[8] Balas, E., Samuelsson, H.: A node covering algorithm. Naval Research Logistics Quarterly **24**(2), 213–233 (June 1977)

[9] Balas, E., Xue, J.: Minimum weighted coloring of triangulated graphs, with application to maximum weight vertex packing and clique finding in arbitrary graphs. SIAM Journal on Computing **20**, 209–221 (1991)

[10] Balas, E., Yu, C.S.: Finding a maximum clique in an arbitrary graph. SIAM Journal on Computing **15**(4), 1054–1068 (1986)

[11] Barker, G., Carlson, D.: Cones of diagonally dominant matrices. Pacific Journal of Mathematics **57**, 15–32 (March 1975)

[12] Benson, S., Ye, Y.: Approximating maximum stable set and minimum graph coloring problems with the positive semidefinite relaxation. Applications and Algorithms of Complementarity (July 2000)

[13] Carraghan, R., Pardalos, P.M.: An exact algorithm for the maximum clique problem. Operations Research Letters **9**(6), 375–382 (1990)

[14] Chudnovsky, M., Robertson, N., Seymour, P., Thomas, R.: The strong perfect graph theorem. Annals of Mathematics (2) **164**(1), 51–229 (2006)

[15] Conforti, M., Fiorini, S., Huynh, T., Weltge, S.: Extended formulations for stable set polytopes of graphs without two disjoint odd cycles. Mathematical Programming **192**(1), 547–566 (2022)

[16] Eschen, E.M., Wang, X.: Algorithms for unipolar and generalized split graphs. Discrete Applied Mathematics **162**, 195–201 (2014)

[17] Faenza, Y., Oriolo, G., Stauffer, G.: Separating stable sets in claw-free graphs via padberg-rao and compact linear programs. In: Proceedings of the 2012 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA). pp. 1298–1308 (2012)

[18] Faenza, Y., Oriolo, G., Stauffer, G.: Solving the weighted stable set problem in claw-free graphs via decomposition. Journal of the ACM **61**(4) (July 2014)

[19] Friden, C., Hertz, A., de Werra, D.: Tabaris: An exact algorithm based on tabu search for finding a maximum independent set in a graph. Computers and Operations Research **17**(5), 437–445 (1990)

[20] Fulkerson, D.R., Gross, O.A.: Incidence matrices and interval graphs. Pacific Journal of Mathematics **15**(3), 835 – 855 (1965)

[21] Garey, M.R., Johnson, D.S.: " strong " np-completeness results: Motivation, examples, and implications. Journal of the ACM **25**(3), 499–508 (July 1978)

[22] Garstka, M., Cannon, M., Goulart, P.: COSMO: A conic operator splitting method for convex conic problems. Journal of Optimization Theory and Applications **190**(3), 779–810 (2021). https://doi.org/10.1007/s10957-021-01896-x, `https://doi.org/10.1007/s10957-021-01896-x`

[23] Gavril, F.: Algorithms for a maximum clique and a maximum independent set of a circle graph. Networks **3**(3), 261–273 (1973)

[24] Gavril, F.: Algorithms on circular-arc graphs. Networks **4**(4), 357–369 (1974)

[25] Gavril, F.: Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph. SIAM J. Comput. **1**(2), 180–187 (Jun 1972)

[26] Ge, D., Huangfu, Q., Wang, Z., Wu, J., Ye, Y.: Cardinal Optimizer (COPT) user guide. https://guide.coap.online/copt/en-doc (2023)

[27] Goemans, M.X., Williamson, D.P.: Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. Journal of the ACM **42**(6), 1115–1145 (Nov 1995)

[28] Grötschel, M., Lovász, L., Schrijver, A.: The ellipsoid method and its consequences in combinatorial optimization. Combinatorica **1**(2), 169–197 (1981)

[29] Grötschel, M., Lovász, L., Schrijver, A.: Geometric Algorithms and Combinatorial Optimization. Springer Berlin Heidelberg, Berlin, Heidelberg (1993)

[30] Grötschel, M., Lovász, L., Schrijver, A.: Polynomial algorithms for perfect graphs. In: Topics on Perfect Graphs, North-Holland Mathematics Studies, vol. 88, pp. 325–356. North-Holland (1984)

[31] Gupta, U.I., Lee, D.T., Leung, J.Y.: Efficient algorithms for interval graphs and circular-arc graphs. Networks **12**(4), 459–467 (1982)

[32] Hsu, W.l., Ikura, Y., Nemhauser, G.L.: A polynomial algorithm for maximum weighted vertex packings on graphs without long odd cycles. Mathematical Programming **20**(1), 225–232 (1981)

[33] Johnson, D.S., Trick, M.A.: Cliques, Coloring, and Satisfiability, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 26. American Mathematical Society (1996)

[34] Karp, R.M.: Reducibility among Combinatorial Problems, pp. 85–103. Springer US, Boston, MA (1972)

[35] Lawler, E.L.: A procedure for computing the k best solutions to discrete optimization problems and its application to the shortest path problem. Management Science **18**(7), 401–405 (1972). https://doi.org/10.1287/mnsc.18.7.401, `https://doi.org/10.1287/mnsc.18.7.401`

[36] Leskovec, J., Krevl, A.: SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data` (June 2014)

[37] Liang, Y., Dhall, S., Lakshmivarahan, S.: On the problem of finding all maximum weight independent sets in interval and circular-arc graphs. In: [Proceedings] 1991 Symposium on Applied Computing. pp. 465–470 (1991). https://doi.org/10.1109/SOAC.1991.143921

[38] Lovasz, L.: On the shannon capacity of a graph. IEEE Transactions on Information Theory **25**(1), 1–7 (1979)

[39] Lovász, L., Schrijver, A.: Cones of matrices and set-functions and 0–1 optimization. SIAM Journal on Optimization **1**(2), 166–190 (1991)

[40] Mannino, C., Sassano, A.: An exact algorithm for the maximum stable set problem. Computational Optimization and Applications **3**(3), 243–258 (1994)

[41] Marino, R., Buffoni, L., Zavalnij, B.: A short review on novel approaches for maximum clique problem: from classical algorithms to graph neural networks and quantum algorithms (2024), `https://arxiv.org/abs/2403.09742`

[42] McDiarmid, C., Yolov, N.: Random perfect graphs. Random Structures & Algorithms **54**(1), 148–186 (2019)

[43] Monteiro, R.D.C., Sujanani, A., Cifuentes, D.: A low-rank augmented lagrangian method for large-scale semidefinite programming based on a hybrid convex-nonconvex approach (2024), `https://arxiv.org/abs/2401.12490`

[44] Muir, C., Toriello, A.: Dynamic node packing. Mathematical Programming **196**(1), 875–906 (2022)

[45] Murty, K.G.: Letter to the editor—an algorithm for ranking all the assignments in order of increasing cost. Operations Research **16**(3), 682–687 (1968). https://doi.org/10.1287/opre.16.3.682, `https://doi.org/10.1287/opre.16.3.682`

[46] Nemhauser, G.L., Trotter, L.E.: Vertex packings: Structural properties and algorithms. Mathematical Programming **8**(1), 232–248 (1975)

[47] Nemhauser, G., Sigismondi, G.: A strong cutting plane/branch-and-bound algorithm for node packing. Journal of Operational Research Society **43**, 443–457 (05 1992)

[48] Nesterov, Y., Nemirovskii, A.: Interior-Point Polynomial Algorithms in Convex Programming. Society for Industrial and Applied Mathematics (1994)

[49] Okamoto, Y., Uno, T., Uehara, R.: Counting the number of independent sets in chordal graphs. Journal of Discrete Algorithms **6**(2), 229–242 (2008). https://doi.org/https://doi.org/10.1016/j.jda.2006.07.006, `https://www.sciencedirect.com/science/article/pii/S1570866707000330`, selected papers from CompBioNets 2004

[50] Pardalos, P.M., Rodgers, G.P.: A branch and bound algorithm for the maximum clique problem. Computers and Operations Research **19**(5), 363–375 (1992)

[51] Prosser, P.: Exact algorithms for maximum clique: A computational study. Algorithms **5**(4), 545–587 (2012)

[52] Prömel, H.J., Steger, A.: Almost all Berge Graphs are Perfect. Combinatorics, Probability and Computing **1**(1), 53–79 (1992)

[53] The Sage Developers: SageMath, the Sage Mathematics Software System (Version 10.4) (2024), `https://www.sagemath.org`

[54] Walteros, J.L., Buchanan, A.: Why Is Maximum Clique Often Easy in Practice? Operations Research **68**(6), 1866–1895 (November 2020)

[55] Wu, Q., Hao, J.K.: A review on algorithms for maximum clique problems. European Journal of Operational Research **242**(3), 693–709 (2015)

[56] Ye, Y.: Gset dataset of random graphs. `www.cise.ufl.edu/research/sparse/matrices/Gset/`, accessed: 2023-10-25

[57] Yildirim, E.A., Fan-Orzechowski, X.: On Extracting Maximum Stable Sets in Perfect Graphs Using Lovász's Theta Function. Computational Optimization and Applications **33**(2), 229–247 (2006)

A Additional proofs

Lemma A.1. *Given fixed $Q \in \mathbb{S}^n_+$, $q \in \mathbb{R}^n$, there exists $\varphi > 0$ such that $\varphi qq^\top \preceq Q$ if and only if $q \in \text{Range}(Q)$.*

Proof. Suppose there exists $\varphi > 0$ such that $\varphi qq^\top \preceq Q$. For contradiction, assume $q \notin \text{Range}(Q)$; then $q = a + b$, where $a \in \text{Range}(Q)$ and $b \in \text{Null}(Q) \setminus \{0\}$ by the orthogonal decomposition. Then

$$0 < \varphi \|b\|_2^2 = \varphi b^\top qq^\top b \leq b^\top Qb = 0,$$

which gives a contradiction.

Suppose $q \in \text{Range}(Q)$; then we can write $q = Qy$ for some y. By the singular value decomposition, there is an orthonormal matrix U and a diagonal matrix D such that $Q = UDU^\top$. Without loss of generality, assume $D_{11} \geq \ldots D_{rr} > 0$, for $r := \text{rank}(Q)$. Thus, we have

$$U^\top qq^\top U = DU^\top yy^\top UD,$$

where $[U^\top qq^\top U]_{ij} = 0$ if $i > r$ or $j > r$. Hence, $U^\top qq^\top U \preceq \varphi D$ for some $\varphi > 0$, and $\varphi^{-1} qq^\top \preceq UDU^\top$. □

Graph				Solution value				Time (s)			
Name	$\|N\|$	$\|E\|$	ϑ	Alg.3	BY_{avg}	BY_{best}	WB	SDP	Alg.3	BY_{total}	WB
G1	800	19176	145.03	86	34.61	54		144.62	5.26	0.95	
G2	800	19176	145.35	86	34.38	54		144.35	2.65	1.26	
G3	800	19176	145.3	86	34.25	57		136.46	2.83	1.01	
G4	800	19176	145.37	85	34.87	54		142.12	2.29	0.86	
G5	800	19176	145.36	85	34.91	53		139.21	2.52	0.73	
G6	800	19176	145.03	86	34.1	55		147.05	2.49	0.76	
G7	800	19176	145.35	86	34.43	54		148.82	3.1	1.27	
G8	800	19176	145.3	86	34.21	56		144.48	3.21	0.75	
G9	800	19176	145.37	85	34.58	54		142.7	2.42	0.8	
G10	800	19176	145.36	85	34.83	53		137.69	2.39	0.75	
G11	800	1600	400.02	400	400	400	400	2.21	1.55	0.14	0.27
G12	800	1600	400.01	400	400	400	400	0.13	0.11	0.09	0.28
G13	800	1600	398.42	384	381.42	384		1.37	0.17	0.09	
G14	800	4694	279	279	275.45	279		42.65	0.34	0.17	
G15	800	4661	283.85	283	270.32	282		220.98	26.45	0.18	
G16	800	4672	285.16	285	266.85	284		220.98	81.01	0.24	
G17	800	4667	286.23	286	271.54	285		191.85	24.56	0.18	
G18	800	4694	279	279	275.45	279		41.01	0.27	0.17	
G19	800	4661	283.85	283	269.63	283		226.4	27.16	0.19	
G20	800	4672	285.16	285	267.93	285		224.62	78.44	0.19	
G21	800	4667	286.23	286	269.91	286		191.94	24.94	0.21	
G22	2000	19990	578.51	410	238.82	305		377.25	36.83	3.74	
G23	2000	19990	577.93	415	237.18	299		378.67	33.63	4.21	
G24	2000	19990	580.01	411	239.74	308		378.68	35.27	3.78	
G25	2000	19990	578.09	406	237.85	301		379.84	33.88	4.08	
G26	2000	19990	577.99	411	238.09	299		368.94	35.1	3.86	
G27	2000	19990	578.51	410	238.37	309		366	33.38	6.37	
G28	2000	19990	577.93	415	238.72	300		367.88	34.05	3.95	
G29	2000	19990	580.01	411	239.24	303		369.62	37.82	3.77	
G30	2000	19990	578.09	406	237.12	301		370.36	38.12	4.42	
G31	2000	19990	577.99	411	237.74	301		371.64	34.09	4.02	
G32	2000	4000	1000	1000	1000	1000	1000	0.35	0.64	0.53	20.95
G33	2000	4000	996.04	960	945.7	960		191.85	304.3	0.55	
G34	2000	4000	1000	1000	1000	1000	1000	0.27	0.56	0.52	20.44
G35	2000	11778	718.27	718	693.53	716		378.97	0.89	1.13	
G36	2000	11766	696.03	695	671.48	695		586.57	101.99	1.11	
G37	2000	11785	708.02	708	687.27	707		306.7	0.95	1.16	
G38	2000	11779	716.02	716	694.83	715		298.5	0.91	1.11	
G39	2000	11778	718.27	718	693.53	716		372.92	1.17	1.12	
G40	2000	11766	696.03	695	670.97	695		571.55	103.25	1.1	
G41	2000	11785	708.02	708	687.27	707		299.61	1.04	1.13	
G42	2000	11779	716.02	716	694.83	715		306.85	1.22	1.11	
G43	1000	9990	280.62	199	114.89	151		42.47	6.6	0.72	
G44	1000	9990	280.58	206	117.66	152		42.35	4.72	0.68	
G45	1000	9990	280.19	198	115.24	149		41.16	5.03	1.03	
G46	1000	9990	279.84	199	115.75	155		40.47	7.43	0.66	
G47	1000	9990	281.89	203	117.81	152		40.65	4.91	0.66	
G48	3000	6000	1500	1500	1500	1500	1500	0.48	142.93	1.18	23.40
G49	3000	6000	1500	1500	1500	1500	1500	0.47	0.86	1.17	23.99
G50	3000	6000	1494.06	1440	1418.94	1440		226.4	1104.64	1.27	
G51	1000	5909	349.01	349	333.54	348		184.94	0.38	0.29	
G52	1000	5916	348.43	348	332.37	348		220.98	2.16	0.36	
G53	1000	5914	348.39	346	324.99	346		255.35	0.4	0.29	
G54	1000	5916	341.01	341	330.08	341		120.39	23.68	0.29	
G55	5000	12498	2324.17	2172	1877.34	2060		120.82	19.77	3.73	
G56	5000	12498	2324.17	2172	1877.34	2060		124.67	18.9	3.7	
G57	5000	10000	2500	2500	2500	2500		0.96	1.87	3.53	
G58	5000	29570	1782.62	1782	1714.53	1778		1291.86	8.22	7.62	
G59	5000	29570	1782.62	1782	1714.53	1778		1286.69	7	6.9	
G60	7000	17148	3265.04	3056	2643.67	2883		251.85	47	7.53	
G61	7000	17148	3265.04	3056	2643.67	2883		246.22	47.75	8.32	
G62	7000	14000	3500	3500	3500	3500		2.89	3.78	6.93	
G63	7000	41459	2493.42	2486	2381.93	2484		1831.07	21.6	13.96	
G64	7000	41459	2493.42	2486	2381.93	2484		6426.76	21.35	14.05	
G65	8000	16000	4000	4000	4000	4000		2.97	199.94	8.53	
G66	9000	18000	4500	4500	4500	4500		4.46	239.33	11.19	
G67	10000	20000	5000	5000	5000	5000		3.32	291.93	13.94	
G70	10000	9999	6077.24	6077	6077	6077		0.05	<0.01	<0.01	
G72	10000	20000	5000	5000	5000	5000		4.61	296.21	14.32	
G77	14000	28000	7000	7000	6999.5	7000		5.34	21.42	28.55	
G81	20000	40000	10000	10000	10000	10000		11.28	1046.98	56.23	

Table 2: Same experiments as in Table 1, using graphs in the GSet dataset [56]. We do not include α, as this number is not included for this dataset.

Name	Graph $\|N\|$	$\|E\|$	ϑ	α	Solution value Alg.3	BY_{avg}	BY_{best}	WB	Time (s) SDP	Alg.3	BY_{total}	WB
DIMACS10												
uk	4824	6837	2286.98		2173	2066.26	2115		53.32	18.33	2.68	
data	2851	15093	690.01		683	621.34	674		118	5.07	2.37	
fe_4elt2	11143	32818	3631.78		3544	2780.99	3239		371.03	93.07	24	
vsp_p0291_seymourl_iiasa	10498	53868	6301.12	6301	6301	6232.51	6301		753.19	6.57	20.62	
cti	16840	48232	8198.1		8083	7798.45	8080		3420.47	152.84	68.99	
fe_sphere	16386	49152	5462.00	5462	5462	4279.16	5462		49.70	2575.14	51.34	
cs4	22499	43858	9738.86		8987	8099.92	8273		534.88	2221.8	84.03	
hi2010	25016	62063	11022.07		11012	10743.45	10926		1464.83	81.59	21.19	
ri2010	25181	62875	10819.96		10792	10460.01	10653		1281.81	1307.78	82.03	
vt2010	32580	77799	15127.13		15118	14826.15	14980		1564.35	1287.42	103.47	
nh2010	48837	117275	22890.74		22878	22452.91	22687		2467.79	3164.02	170.91	
delaunay_n14	16384	49122	5230.5		5132	4141.92	4503		548.09	229.26	54.14	
SNAP												
ca-CondMat	23133	93497	9612.17	9612	9612	9477.4	9585		8328.38	29.85	47.59	
p2p-Gnutella31	62586	76950	47974	47974	47974	47974	47974	47974	0.13	<0.01	<0.01	<0.01
Oregon-2	11806	32730	9889	9889	9889	9888.94	9889	9889	2.53	5.01	0.39	<0.01
p2p-Gnutella25	22687	30751	17116	17116	17116	17116	17116	17116	0.02	<0.01	<0.01	<0.01
p2p-Gnutella24	26518	35828	19872	19872	19872	19872	19872	19872	0.01	<0.01	<0.01	<0.01
as-caida_G_001	31379	32955	29037	29037	29037	29037	29037	29037	1.13	<0.01	<0.01	<0.01
p2p-Gnutella30	36682	48507	28094	28094	28094	28094	28094	28094	0.1	<0.01	<0.01	<0.01

Table 3: Same experiments as in Table 1, using graphs from the DIMACS10 and SNAP datasets [7, 36]

Name	Graph $\|N\|$	$\|E\|$	ϑ	Solution value Alg.3	BY_{avg}	BY_{best}	Time (s) SDP	Alg.3	BY_{total}
MANN-a27	378	702	938	938	937.08	938	7.62	8.79	0.49
MANN-a45	1035	1980	2596	2596	814.16	1791	21.00	18.82	0.28
MANN-a81	3321	6480	8631	8631	2704.07	5939	43.75	109.05	4.31
MANN-a9	45	72	109	109	109	109	0.02	0.01	< 0.01
brock200-1	200	5066	173.47	144	55.23	106	5.16	1.49	0.06
brock200-2	200	10024	89.68	71	21.55	56	19.3	1.04	0.1
brock200-3	200	7852	118.69	93	32.55	76	11.85	1.14	0.07
brock200-4	200	6811	128.47	103	36.69	95	8.83	1.36	0.06
brock400-1	400	20077	246.12	161	51.3	102	151.81	5.06	0.34
brock400-2	400	20014	245.5	166	49	99	146.95	5.4	0.49
brock400-3	400	20119	250.03	169	53.28	111	130.22	5.86	0.34
brock400-4	400	20035	243.72	152	49.13	106	150.03	9.02	0.34
brock800-1	800	112095	259.25	122	27.79	83	83.08	25.17	5.38
brock800-2	800	111434	260.98	112	28.85	86	83.92	28.51	3.88
brock800-3	800	112267	266.33	143	30.09	75	86.77	25.87	4.31
brock800-4	800	111957	265.39	123	29.14	76	83.73	25.94	4.06
c-fat200-1	200	18366	94	94	94	94	2	9.6	0.76
c-fat200-2	200	16665	147	147	146.27	147	2.44	14.83	0.28
c-fat200-5	200	11427	374	374	374	374	14.84	0.2	0.13
c-fat500-10	500	78123	732	732	727.59	732	55.87	15.27	2.67
c-fat500-1	500	120291	97	97	96.81	97	5.2	5.71	2.39
c-fat500-2	500	115611	165	165	165	165	29.64	199.23	2.71
c-fat500-5	500	101559	410	410	409.55	410	230.39	1.43	2.03
c1000.9	1000	49421	759.58	464	140.96	272	1697.06	1400.01	9.85
c125.9	125	787	218.98	195	138.33	189	8.02	8.61	0.55
c2000.5	2000	999164	280.52	99	15.35	66	33155.19	879.2	227.15
c2000.9	2000	199468	1105.27	344	48.83	242	72603.8	3033.9	22.22
c250.9	250	3141	341.49	271	146.54	233	1.67	2.7	0.07
c4000.5	4000	3997732	395.4	106	13.75	57	462610.07	3140.67	1181.26
c500.9	500	12418	523.52	365	143.28	270	50.33	60.73	0.46
dsjc1000_5	1000	249674	200.04	92	18.05	59	109.71	813.45	25.78
dsjc500_5	500	62126	146.51	96	18.95	57	28.27	173.71	1.6
gen200_p0.9_44	200	1990	291.78	246	142.57	229	1.28	18.26	0.06
gen200_p0.9_55	200	1990	338	338	338	338	1.17	11.09	0.17
gen400_p0.9_55	400	7980	423.61	324	177.53	289	22.25	333.86	0.32
gen400_p0.9_65	400	7980	446.12	352	168.08	282	21.6	218.57	0.38
gen400_p0.9_75	400	7980	455.6	320	187.45	409	21.11	230.69	0.34
hamming10-2	1024	5120	2800	2800	2792.28	2800	11.42	23.95	2.33
hamming10-4	1024	89600	364.71	285	122.63	238	9918.86	45700.81	8.57
hamming6-2	64	192	173	173	173	173	8.59	3.94	0.08
hamming6-4	64	1312	38.1	37	21.05	37	0.48	0.1	< 0.01
hamming8-2	256	1024	737	737	723.47	737	0.21	0.32	0.02
hamming8-4	256	11776	123	123	123	123	32.29	4.93	0.16
johnson16-2-4	120	1680	77	77	77	77	0.5	0.35	0.02
johnson32-2-4	496	14880	157	157	148.84	157	91.22	148.94	0.9
johnson8-2-4	28	168	38	38	38	38	0.04	0.02	< 0.01
johnson8-4-4	70	560	105	105	105	105	0.14	0.03	< 0.01
keller4	171	5100	95.73	79	31.5	76	3.47	1.55	0.04
keller5	776	74710	234.45	176	44.28	125	1134.35	125.81	3.14
keller6	3361	1026582	538.62	277	64.92	203	212884.82	11605.11	259.51
p-hat1000-1	1000	377247	109.19	64	13.85	47	1641.78	66.42	15.51
p-hat1000-2	1000	254701	335.69	277	121.58	245	2272.97	254.03	12.79
p-hat1000-3	1000	127754	519.58	406	151.7	321	3162.75	77.13	6.62
p-hat1500-1	1500	839327	136.33	56	12.92	52	12036.66	216.02	59.58
p-hat1500-2	1500	555290	460.14	292	146.28	297	10208.36	284.65	104.05
p-hat1500-3	1500	277006	697.66	332	203.96	467	36496.32	3961.68	26.99
p-hat300-3	300	11460	245.55	221	97.72	187	44.24	12.05	0.68
p-hat500-1	500	93181	82.56	58	15.74	58	202.27	9.87	1.89
p-hat500-2	500	61804	239.35	220	110.89	191	554.96	19.46	1.22
p-hat500-3	500	30950	353.66	317	149.25	303	659.87	8.71	0.62
p-hat700-1	700	183651	94.58	59	15.44	46	526.38	25.86	4.62
p-hat700-2	700	122922	300.27	250	117.79	230	890.9	36.62	3.89
p-hat700-3	700	61640	436.92	367	33.74	248	2497.79	108.12	2.6
san1000	1000	249000	114.51	87	19.82	76	2698.35	125.21	10.73
san200-0-7-1	200	5970	159	159	159	159	5.39	0.58	0.07
san200-0-7-2	200	5970	125.31	101	63.13	108	6.6	4.49	0.05
san200-0-9-1	200	1990	398	398	398	398	0.78	0.42	0.02
san200-0-9-2	200	1990	295.59	251	163.16	268	0.89	2.38	0.02
san200-0-9-3	200	1990	286.18	246	158.95	241	0.88	2.3	0.04
san400-0-5-1	400	39900	83.17	72	23.88	63	849.72	11.59	0.61
san400-0-7-1	400	23940	210.23	173	62.9	155	280.5	6.05	0.58
san400-0-7-2	400	23940	191.04	129	47.58	120	281.6	6.61	0.42
san400-0-7-3	400	23940	171.06	128	43.33	96	281.84	10.56	0.4
san400-0-9-1	400	7980	489.08	386	282.29	489	17.21	8.33	0.18
sanr200-0-7	200	6032	149.63	121	44.62	112	5.78	1	0.05
sanr200-0-9	200	2037	298.27	262	136.82	217	0.97	1.74	0.02
sanr400-0-5	400	39816	129.77	75	20.16	57	15.4	3.49	0.61
sanr400-0-7	400	23931	211.66	134	43.06	97	240.72	5.37	0.45

Table 4: Comparison of Algorithm 3 with Benson-Ye (BY) on the DIMACS benchmark instances in the order of Table 1, with weights independently sampled from Uniform($[10]$). All instances are complemented first; $|E|$ is the number of edges after taking the complement. Algorithm 3 reaches optimality or the best known bound for bolded instances. We run the randomized BY method $|N|$ times and report the average and best value obtained.