SAPIENZA
UNIVERSITÀ DI ROMA

Sapienza University of Rome

DIPARTIMENTO DI INGEGNERIA INFORMATICA, AUTOMATICA E GESTIONALE
PhD in Automatic Control, Bioengineering and Operations Research (ABRO)

DOCTORAL THESIS

On Semidefinite Lift-and-Project relaxations of Combinatorial Optimization Problems

Candidate
Federico Battista

Supervisor
Prof. Marianna De Santis

Co-Supervisors
Prof. Fabrizio Rossi
Prof. Stefano Smriglio

Academic Year MMXIX-MMXXII (XXXV cycle)

Per aspera ad astra...

Abstract

Finding the stability and the chromatic number of a graph are two among the fundamental problems in combinatorial optimization. Given a graph, the first calls for a stable set of maximum cardinality, i.e. a subset of vertices such that no two are adjacent; the latter asks for a partition of the nodes into the minimum number of stable sets (i.e. colors).

Both the stable set and graph coloring problems are well-known to be NP-hard, hence no polynomial time algorithm to solve them exactly is expected to exists unless P=NP. Thus, the study of strong relaxations of these two problems is a well-researched topic. In particular, the Lovász theta function $\theta(G)$ provides at the same time a good upper bound on the stability number of a graph G and a lower bound on the chromatic number of its complement. It can be computed in polynomial time by solving a semidefinite program, which in addition turns often out to be fairly tractable in practice. As a consequence, $\theta(G)$ achieves a hard-to-beat trade-off between computational effort and strength of the bound. Hierarchies of relaxations to strengthen $\theta(G)$ both towards the stability and chromatic number have been documented, but in general such improvements come at a heavy computational burden with off-the-shelf SDP algorithms and require highly specialized methods to be addressed.

In the last decades, Lift-and-Project methods have gained a lot of attention, being able to generate strong relaxations for combinatorial optimization problems. In particular, starting from any linear relaxation Lovász and Schrijver's Lift-and-Project framework generates a semidefinite relaxation. Its application to the fractional stable set polytope showed its potential, producing bounds stronger than $\theta(G)$ but in general they come at a nontrivial computational cost.

In this thesis we introduce a new semidefinite relaxation for the stable set problem obtained by the lifting of a more compact linear formulation than the classical one. Then, we characterize some classes of valid inequalities for the stable set polytope which are implied by our proposal. We then discuss how to face the computational burden arising from these semidefinite programs by the employment of a general purpose solver for SDPs.

Despite Lift-and-Project applications have been widely studied on the Stable Set problem, to the best of our knowledge none on the Graph Coloring problem have been presented. We investigate its employment in this problem, showing that the resulting SDP can yield bounds above the fractional chromatic number, a remarkable threshold not so straightforward to cross with semidefinite programming.

Although interior-point methods achieve good accuracy in reasonable time for small and medium size SDPs, their scalability to large instances is often compromised by memory requirements. On the other hand, Alternating Direction Methods of Multipliers currently represent the most popular

first-order alternatives, being suited to scale to much larger semidefinite programs. This of course at some cost in accuracy, that should be correctly addressed when bounding the optimal solution of some combinatorial optimization problem. In this work we focus on an ADMM designed for SDPs in standard form and extend it to deal with inequalities. Moreover, we report different methods to compute a valid bound on the optimal value of the SDP starting from a medium accuracy solution and we discuss the employments of these methodologies within ADMMs.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

In a Combinatorial Optimization (CO) problem, we want to find a set from a finite family of subsets $\mathcal{F}$ (i.e. *feasible solutions*) made up of elements from a finite set $E = \{e_1, \ldots, e_n\}$, which is the best under a given cost function.

There are many different real-world problems falling under such a definition and a wide literature on them: covering, packing, scheduling, assignment, shortest path, traveling salesman, etc.

One of the most important methods in combinatorial optimization is to represent each feasible solution of the problem by a 0-1 vector and then describe the convex hull K of the solutions by a system of linear inequalities. In a perfect world, this would allow one to compute the optimum of any linear objective function in polynomial time by the usage of Linear Programming (LP) techniques, but in most cases such tight description of integral solutions has an exponential size (in the number of inequalities and/or number of variables) and this causes such an object to be hard to handle in general. Therefore, a customary in the literature is to try and find "good" *relaxations* of the convex hull of many CO problems, i.e. descriptions of a larger, but easier to handle convex region $P \supseteq K$ whose integral solutions coincide with the latter. By doing this, apart from a few very special cases, we pay a price: optimizing a linear function over P yields only a bound on the optimal value of the original problem.

The *stable set* problem is one of the fundamental combinatorial optimization problem, which is sometimes referred as *independent set* or *vertex packing problem*.

Maximum Stable Set Problem Let $G = (V, E)$ be a graph. Then a subset of vertices $S \subseteq V$ is called a stable set if no two vertices of S are adjacent. A stable set is called maximum if there is no stable set with larger cardinality in G. The cardinality of a maximum stable set is called stability number of G and denoted by $\alpha(G)$. The maximum stable set problem (MSSP) calls for a stable set of size $\alpha(G)$.

Many results on the MSSP have been presented in literature: Karp [41] proved in 1972 that deciding whether $\alpha(G) \geq k$ or not for a given $k \in \mathbb{N}$ is an NP-complete problem, while Håstad [34] proved that $\alpha(G)$ is hard to approximate within $n^{1-\epsilon}$ for any $\epsilon > 0$. Therefore there is no polynomial time algorithm for solving the stable set problem unless P=NP. The stable set problem is strictly related to another well-established combinatorial optimization problem

Maximum Clique Problem Let $G = (V, E)$ be a graph. Then a subset of vertices $K \subseteq V$ is called a clique if $\{i, j\} \in E$ for all vertices $i, j \in K$. A clique is called maximum if there is no clique with larger cardinality in G. The cardinality of a maximum clique is called clique number of G and denoted by $\omega(G)$. The maximum clique problem (MCP) calls for a clique of size $\omega(G)$.

It is well known that $\alpha(G) = \omega(\bar{G})$, i.e. the stable set problem on G is equivalent to the maximum clique problem on $\bar{G}$, the complement graph of G. These problems have a wide range of applications in practical context, see for example [11, 13, 68]. Numerous approaches to solve or approximate the stable set problem have been proposed. We refer to the paper by Bomze et al. [7] and more recently, Wu and Hao [81] for a survey of exact algorithms (such as explicit and implicit enumerations) and heuristics (such as sequential greedy approaches, local and random searches) for the maximum clique problem and therefore the stable set problem. At last, let us introduce another fundamental CO problem.

Graph Coloring Problem Let $G = (V, E)$ be a graph and let $k \in \mathbb{N}$. Then a k-coloring of the graph is a partition of V into k stable sets. The minimum number k s.t. G has a k-coloring is denoted by $\chi(G)$. The graph coloring problem (GCP) calls for finding $\chi(G)$.

The coloring problem was also proven to be NP-Hard by Karp [41]. For more information about this topic, we refer the reader to the survey of Malaguti and Toth [55], for example.

The first and straightforward approach to obtain a "natural" relaxation of a CO problem is to formulate it as an *integer linear program* (ILP) in which the vector of variables must satisfy the (non-convex) constraint $x \in \{0, 1\}^n$ and then replace this constraint with $x \in [0, 1]^n$. On the one hand, linear relaxations are relatively fast to compute using simplex-based algorithms or the interior point methods, on the other hand the resulting bounds are frequently weak. As a result, a significant amount of time must be spent in enumeration if one wishes to solve instances to optimality (see e.g., Rossi and Smriglio [67]).

Another kind of relaxations arise from Semidefinite Programming (SDP), whose interest has grown in the last two decades. From a high-level of perspective, semidefinite programming can be considered a generalization of linear programming since they both belongs to the wider field of convex optimization: in LP the vector variable x is restricted to the cone of non-negative real vectors $\mathbb{R}^n_+$ while in SDP, a symmetric matrix variable X is restricted to the cone of symmetric positive semidefinite matrices $\mathcal{S}_n^+$, also denoted by $X \succeq 0$. The first success of SDP in the field of CO dates back to 1979 when Lovász in his seminal paper [52] introduced the Lovász theta function $\theta(G)$ of a graph G. He proved the following sandwich theorem

$$\alpha(G) \leq \theta(G) \leq \chi(\bar{G}),$$

giving at the same time an upper bound for the stability number $\alpha(G)$ and a lower bound on the chromatic number $\chi(\bar{G})$ of the complement graph for every graph G. Some years later, Goemans and Williamson [25] presented a semidefinite relaxation for the Max-Cut problem which was proven to yield an approximation ratio of 0.878. Due to the numerous applications arose from this field along with the possibility to formulate strong relaxations for NP-Hard problems, the demand of efficient algorithms for solving Semidefinite Programs increased. The peculiar property of SDP is that one can solve a semidefinite program to arbitrary precision in polynomial time [76] using interior point methods [62]. For small and medium scale problems, these methods are a well-established tool but their memory requirements are too demanding for large SDPs. On the other hand, augmented Lagrangian based approaches such as the boundary point method in [56] and later elaborated by Cerulli, De Santis, Gaar and Wiegele [14], represent a valid alternative since, as first-order methods they allow to scale better on large instances.

A wide range of approaches have been developed in literature having the scope to tighten both linear and semidefinite relaxations: the general idea behind is to find *valid* (*linear*) *inequalities* for the convex hull K of the problem concerned but violated by its relaxation P, so that such an inequality "cuts" part of the latter, as a matter of fact we refer to these kind of inequalities as *cutting planes* or simply *cuts*. One of the first general approaches was introduced by Gomory, exploiting integrality properties of the problem with the so-called Chvátal-Gomory cuts, see for example [80]. Extensive research has been done for finding valid (linear) inequalities for many convex hulls arising from specific combinatorial optimization problems such as the stable set for which we refer to Padberg [63], for example. Another popular method which will be part of the focus in this thesis, is to try to represent K as the projection of another convex set Q lying in a higher (but still polynomial) dimensional space. The idea behind this is that the projection of Q may have more facets than Q itself. Hence it might be that even if the description of K has an exponential size, it can be obtained as the projection of a set Q whose description in that dimensional space is of polynomial size (we say that Q admits a *compact representation* in such a space). Several methods have been developed for constructing projection representations for general 0-1 problems: Sherali and Adams [71], Balas et al. [2], Lasserre [44] and Lovász and Schrijver [53]. Our focus will be on the latter and it will be addressed in Chapters 4 and 5. A detailed comparison of these methods have been presented in Laurent [46]. A common feature is the construction of a hierarchy of relaxations, which typically involve several levels, each of which gets increasingly tighter to the exact solution, but the computational effort grows accordingly. More recently, a related work specialized to the stable set, graph coloring and max-cut problems was introduced by Adams, Anjos, Rendl and Wiegele [1] and revised in Gaar and Rendl [20].

To summarize, this thesis deals with the following new studies:

- A new Semidefinite relaxation obtained by the application of Lovász and Schrijver's Lift-and-Project operator to a compact linear formulation for the Maximum Stable Set problem.
- An innovative application of the $M_+(\cdot)$ operator to the Graph Coloring problem in order to obtain a new Semidefinite relaxation.
- The implementation development of an Alternating Direction Method of Multipliers capable of dealing with inequality constraints, addressing the issue of computing valid bounds on the optimal solution of Semidefinite programs. Furthermore, a collection of benchmarks are built up and experiment's results are reported.

Chapter 2

Preliminaries

For the sake of this work's self-containment, in this Chapter we report some basic definitions and results on cones and polyhedra which will be useful later. Moreover, in Section 2.2 a brief introduction on Semidefinite Programs including its duality theory is given.

For a detailed treatment, we refer the reader to Bertsimas and Tsitsiklis' text-book [4] for Section 2.1 and to the surveys of Helmberg [35] or Laurent and Rendl [47] for Section 2.2.

2.1 Cones and Polyhedra

Let us start with the formal definition of a *polyhedron* [4]:

Definition 2.1. *A polyhedron P is a set that can be described in the form*

$$P = \{x \in \mathbb{R}^n \mid Ax \leq b\},$$

with $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$.

If a polyhedron P is bounded, then we refer to P as a *polytope*. Polyhedra are always closed sets. Moreover, they belong to the more general class of *convex* sets.

Definition 2.2. *A set $S \in \mathbb{R}^n$ is convex if for any $x, y \in S$ and any $c \in [0, 1]$, $cx + (1 - c)y \in S$ holds.*

The geometric interpretation of this definition is that S is convex if and only if for all $x, y \in S$ all the points belonging to the line connecting x and y are also in S. Given a set of vectors $S = \{x_1, \ldots, x_k \in \mathbb{R}^n\}$ and a set of scalars $c_1, \ldots, c_k \in \mathbb{R}^n_+$ with $\sum_{i=1}^k c_i = 1$ then the vector

$$\sum_{i=1}^{k} c_i x_i,$$

is a *convex combination* of vectors in S. Then the *convex hull* of S, denoted by $conv(S)$ is the set of all convex combinations of vectors in S. When addressing combinatorial optimization problems with linear programming, for example, we describe the solutions of these problems as 0-1 vectors contained in $Q_n = [0,1]^n$, the n-dimensional unit cube. The convex hull of (a subset of) 0-1 points contained in Q_n is always a polytope.

Let $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ be a polyhedron. Then an inequality $c^\top x \leq \delta$ is *valid* for P if it is satisfied by all points in P, i.e. if $c^\top \bar{x} \leq \delta$ for all $\bar{x} \in P$. Then, the hyperplane $H = \left\{x \mid c^\top x = \delta\right\}$ is called a *supporting hyperplane* of P if $c^\top x \leq \delta$ is valid for P and H intersects P. Hence, we can give the following definition of *faces*:

Definition 2.3. *A face F of a polyhedron P is its intersection with a supporting hyperplane of P. In other words a face is a set of the form*

$$F = P \cap \left\{x \mid c^\top x = \delta\right\},$$

where $c^\top x \leq \delta$ is a valid inequality for P.

Given a polyhedron P we are usually interested in identifying its *facets*.

Definition 2.4. *F is a facet of P if and only if the following hold:*

1. *F is a face of P;*
2. $\dim(F) = \dim(P) - 1$.

In other words, facets are faces of P of maximum dimension.

Now let us introduce the definition of *cones*:

Definition 2.5. *A set $C \subseteq \mathbb{R}^n$ is a cone if $0 \in C$ and for every $x \in C$ and $c \geq 0$, cx also belongs to C. In other terms, C is a cone if and only if it contains the origin and, for every $x \in C \setminus \{0\}$, C contains the half line starting from the origin and passing through x.*

In particular, a *polyhedral cone* is a polyhedron defined by a homogeneous system of inequalities, that is

$$C_P = \{x \in \mathbb{R}^n \mid Ax \geq 0\}.$$

We remark that, any polyhedron $P := \{x \in \mathbb{R}^n \mid Ax \leq b\}$ can be seen in a space of dimension $n+1$ (with an additional variable x_0) as the intersection of a polyhedral cone with the hyperplane $x_0 = 1$.

Cones need not to be closed. If C is a cone in $\mathbb{R}^n$, its *dual* or *polar cone* is C^* and it is defined by

$$C^* = \left\{y \in \mathbb{R}^n \mid y^\top x \geq 0 \text{ for all } x \in C\right\}.$$

The dual cone C^* is the cone spanned by the vectors defining valid inequalities for C. C^* is always a closed set. Moreover, if C is closed, then $(C^*)^* = C$ and we say that C is self dual. Examples of self dual cones are the nonnegative orthant $\mathbb{R}^n_+$ and the cone of positive semidefinite matrices $\mathcal{S}_n^+$.

2.2 Semidefinite Programming

In this section we report some basic definitions and results about semidefinite programming along with duality theory. For a complete survey on SDP we refer the reader to Helmberg [35].

Definition 2.6. *Let $Y \in \mathcal{S}_n$ be a symmetric matrix. Then Y is positive semidefinite if $v^\top Y v \geq 0$ for all $v \in \mathbb{R}^n$ and we write $Y \succeq 0$ or $Y \in \mathcal{S}_n^+$, equivalently.*

There are several equivalent definitions of positive semidefinite matrices, the following proposition collects them together:

Proposition 2.7. *Let $Y \in \mathcal{S}_n$, then the following statements are equivalent:*

1. *Y is positive semidefinite.*
2. *$v^\top Y v \geq 0$ for all $v \in \mathbb{R}^n$.*
3. *All eigenvalues of Y are non-negative.*
4. *$\exists\ U \in \mathbb{R}^{m\times n}$ such that $Y = U^\top U$. For any such U, $\operatorname{rank}(U) = \operatorname{rank}(Y)$.*

It will be useful to remark some properties of positive semidefinite matrices. A *principal submatrix* of a square matrix B is a square submatrix B' obtained by deleting a subset of rows and the corresponding columns from that matrix, then

Proposition 2.8. *Let $Y \in \mathcal{S}_n^+$, then*

1. *Every principal submatrix of Y is positive semidefinite and $\det(Y) \geq 0$.*
2. *$\forall\ i \in [n]$, $\operatorname{diag}(Y) \geq 0$. If $y_{ii} = 0$ for some i, then $y_{ij} = y_{ji} = 0$ for all $j \in [n]$.*
3. *If $A \in \mathbb{R}^{n\times n}$ is non-singular, then $A^\top Y A \succeq 0$.*

Lemma 2.9. *Let $A, B \in \mathcal{S}_n^+$. Then*

1. *$\langle A, B\rangle \geq 0$,*
2. *$\langle A, B\rangle = 0 \iff AB = 0$.*

Now, let C and $A_1, \ldots, A_m$ be matrices in $\mathcal{S}_n$ and $b \in \mathbb{R}^m$, then a Semidefinite Program in its standard notation is defined as follows:

$$\begin{aligned} \min \quad & \langle C, X \rangle \\ \text{s.t.} \quad & \mathcal{A}(X) = b \\ & X \in \mathcal{S}_n^+ \end{aligned} \tag{PSDP}$$

where $\mathcal{A} : \mathcal{S}_n \to \mathbb{R}^m$ is a linear operator defined as

$$\mathcal{A}(X) = \begin{pmatrix} \langle A_1, X \rangle \\ \vdots \\ \langle A_m, X \rangle \end{pmatrix}.$$

The adjoint operator $\mathcal{A}^\top : \mathbb{R}^m \to \mathcal{S}_n$ can be derived through the equation

$$\langle \mathcal{A}(X), y \rangle = \left\langle X, \mathcal{A}^\top(y) \right\rangle, \text{ for all } X \in \mathcal{S}_n, \ y \in \mathbb{R}^m.$$

Therefore,

$$\langle \mathcal{A}(X), y \rangle = \sum_{i=1}^m y_i \langle A_i, X \rangle = \sum_{i=1}^m \langle y_i A_i, X \rangle = \left\langle \sum_{i=1}^m y_i A_i, X \right\rangle = \left\langle \mathcal{A}^\top(y), X \right\rangle$$

and hence,

$$\left\langle X, \mathcal{A}^\top(y) \right\rangle = \sum_{i=1}^m y_i A_i.$$

A fundamental in linear optimization is the concept of duality. In order to derive the dual to 6.1, we are going to use the following well-known

Lemma 2.10. ***minimax inequality*** *[66, Lemma 36.1] Let f be a function from the non-empty product set $C \times D$ to $[-\infty, \infty]$, then*

$$\inf_{v \in D} \sup_{u \in C} f(u, v) \geq \sup_{u \in C} \inf_{v \in D} f(u, v).$$

Now we introduce the vector $y \in \mathbb{R}^m$ to be the *Lagrangian multiplier* (or *dual variable*) for the equations. Then the following is always true:

$$
\begin{aligned}
\inf_{X\in\mathcal{S}_n^+} \{\langle C, X\rangle \mid \mathcal{A}(X) = b\} &= \inf_{X\in\mathcal{S}_n^+} \sup_{y\in\mathbb{R}^m} \langle C, X\rangle - \langle \mathcal{A}(X) - b, y\rangle \\
&\geq \sup_{y\in\mathbb{R}^m} \inf_{X\in\mathcal{S}_n^+} \langle b, y\rangle + \left\langle C - \mathcal{A}^\top(y), X\right\rangle \\
&= \inf\left\{\langle b, y\rangle \mid C - \mathcal{A}^\top(y) \in \mathcal{S}_n^+,\ y \in \mathbb{R}^m\right\}.
\end{aligned}
$$

The first equation holds, since when $\mathcal{A}(X) - b \neq 0$, the maximization over y yields ∞, while in the other case $\langle C, X\rangle$ is attained, i.e.

$$
\sup_{y\in\mathbb{R}^m} \langle C, X\rangle - \langle \mathcal{A}(X) - b, y\rangle = \begin{cases} \langle C, X\rangle & \text{if } \mathcal{A}(X) = b \\ \infty & \text{otherwise.} \end{cases}
$$

The inequality is the application of Lemma 2.10 while for the last equation, the value $\langle b, y\rangle$ is attained when $\left\langle C - \mathcal{A}^\top(y), X\right\rangle \geq 0$ and this means that the first argument has to belong to the cone $\mathcal{S}_n^+$ (by Lemma 2.9), i.e.

$$
\inf_{X\in\mathcal{S}_n^+} \langle b, y\rangle + \left\langle C - \mathcal{A}^\top(y), X\right\rangle = \begin{cases} \langle b, y\rangle & \text{if } C - \mathcal{A}^\top(y) \in \mathcal{S}_n^+ \\ -\infty & \text{otherwise.} \end{cases}
$$

Hence now we can write the dual of (PSDP)

$$
\begin{aligned}
\max \quad & \langle b, y\rangle \\
\text{s.t.} \quad & \mathcal{A}^\top(y) + Z = C \\
& y \in \mathbb{R}^m,\ Z \in \mathcal{S}_n^+.
\end{aligned}
\tag{DSDP}
$$

Here a matrix Z is introduced. We can think of Z as "surplus" matrix variable (in a similar manner it is done for linear programming), in this way the sole constraint in (DSDP) is an equality, while Z only has to be positive semidefinite.

Duality theory is one of the fundamentals in convex optimization. Many concepts valid for LPs can be still applied to semidefinite programs, even if for the latter the theory is more involved and for guaranteeing strong duality they have to fulfill the so-called *Slater condition*. Before introducing the basics for duality theory, we are going to state the following

Definition 2.11. ***(Feasibility)*** *A matrix $X \in \mathcal{S}_n^+$ is feasible for* (6.1) *if $\mathcal{A}(X) = b$ holds. The pair $(y, Z) \in \mathbb{R}^m \times \mathcal{S}_n^+$ is feasible for* (DSDP) *if $\mathcal{A}^\top(y) + Z = C$.*

Definition 2.12. ***(Strict feasibility)*** *A matrix $X \in \mathcal{S}_n^+$ is strictly feasible for* (6.1) *if $\mathcal{A}(X) = b$*

holds. The pair $(y, Z) \in \mathbb{R}^m \times \mathcal{S}_n^+$ *is strictly feasible for* (DSDP) *if* $\mathcal{A}^\top(y) + Z = C$.

Now let X be a primal feasible and (y, Z) a dual feasible solution, then the difference between the objective values of the primal and dual feasible solution is defined as *duality gap*.

Definition 2.13. ***(Duality gap)*** *Given the primal-dual feasible pair* $(X, y, Z) \in \mathcal{S}_n^+ \times \mathbb{R}^n \times \mathcal{S}_n^+$ *the* duality gap *is given by*

$$\langle C, X \rangle - \langle b, y \rangle .$$

Since Z and X are PSD, due to Lemma 2.9 the *duality gap* is always non-negative:

$$\langle C, X \rangle - \langle b, y \rangle = \left\langle \mathcal{A}^\top(y) + Z, X \right\rangle - \langle \mathcal{A}(X), y \rangle = \langle Z, X \rangle \geq 0.$$

This is sufficient to prove that *weak duality* between (PSDP) and (DSDP) holds. As for LP, we can formulate it by the following

Lemma 2.14. ***(Weak duality)*** *Let* $X \in \mathcal{S}_n^+$*,* $y \in \mathbb{R}^m$ *with* $\mathcal{A}(X) = b$ *and* $\mathcal{A}^\top(y) + C \succeq 0$*. Then*

$$\langle C, X \rangle \geq \langle b, y \rangle .$$

A more interesting property is when the duality gap is 0 and the quantities $\langle C, X \rangle$, $\langle b, y \rangle$ are equal, i.e. *strong duality* holds. For strong duality to hold in LPs, it suffices that the primal (or equivalently, the dual) attains its optimum. As shown by the example in Vandenberghe and Boyd [76], in Semidefinite Programming the primal and dual attainment of their optima does not imply strong duality.

Example 2.15. *Consider the following SDP*

$$\begin{aligned} \max \quad & x_{12} \\ s.t. \quad & \begin{pmatrix} 0 & x_{12} & 0 \\ x_{12} & x_{22} & 0 \\ 0 & 0 & 1 + x_{12} \end{pmatrix} \succeq 0. \end{aligned}$$

In order to derive the dual, we rewrite it so that the matrices composing the $\mathcal{A}$ *linear operator become* *more evident:*

$$\begin{aligned} \max \quad & x_{12} \\ s.t. & \left\langle \begin{pmatrix} 0 & -\frac{1}{2} & 0 \\ -\frac{1}{2} & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, X \right\rangle = 1 \\ & \left\langle \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, X \right\rangle = 0 \\ & \left\langle \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}, X \right\rangle = 0 \\ & \left\langle \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, X \right\rangle = 0. \end{aligned}$$

Then the dual problem is

$$\min \;\; y_1 + 0y_2 + 0y_3 + 0y_4 \;\; s.t. \;\; y_1 A_1 + y_2 A_2 + y_3 A_3 + y_4 A_4 - C \succeq 0,$$

or equivalently

$$\begin{aligned} \min \quad & y_1 \\ s.t. & \begin{pmatrix} y_2 & \frac{1-y_1}{2} & y_3 \\ \frac{1-y_1}{2} & 0 & y_4 \\ y_3 & y_4 & y_1 \end{pmatrix} \succeq 0. \end{aligned}$$

By Proposition 2.8, x_{12} must be 0 in order X to be positive semidefinite, thus the optimal value of the primal is 0. On the other hand, in the dual $\frac{1-y_2}{2}$ must equal zero, by the same arguments and therefore the optimal dual value is 1. Then the duality gap is 1 for any primal and dual feasible solution.

A sufficient condition for strong duality to hold in semidefinite programming is given by the *Slater condition.*

Definition 2.16. ***(Slater condition)***
(PSDP) satisfies the Slater condition *if there exists* $X \in \mathcal{S}_n^+$ *s.t.* $\mathcal{A}(X) = b$.
(DSDP) satisfies the Slater condition *if there exists a pair* (y, Z) *with* $Z \in \mathcal{S}_n^+$ *s.t.* $\mathcal{A}^\top(y) + Z = C$.

Then we can state the following

Theorem 2.17. *Denote*

$$p^* = \inf \left\{ \langle C, X \rangle \mid \mathcal{A}(X) = b,\ X \in \mathcal{S}_n^+ \right\}$$

and

$$d^* = \sup \left\{ \langle b, y \rangle \mid \mathcal{A}^\top(y) + Z = C \in \mathcal{S}_n^+ \right\}.$$

- *If (PSDP) satisfies the Slater condition with* p^* *finite, then* $p^* = d^*$ *and this value is attained for (DSDP).*
- *If (DSDP) satisfies the Slater condition with* d^* *finite, then* $d^* = p^*$ *and this value is attained for (PSDP).*
- *If (PSDP) and (DSDP) both satisfy the Slater condition, then* $p^* = d^*$ *is attained for both problems.*

A proof of the previous theorem can be found in Nesterov and Nemirovskii [62] or Rockfellar [66]. For semidefinite problems where strong duality holds we therefore obtain the following necessary and sufficient optimality conditions (also referred as Karush-Kuhn-Tucker conditions).

Proposition 2.18. ***(KKT conditions for SDP)*** *Let strong duality holds for the (PSDP) and (DSDP) pair. Then* (X, y, Z) *is optimal if and only if*

$$\begin{aligned} \mathcal{A}(X) = b,\ X \in \mathcal{S}_n^+ & \qquad \textit{(Primal feasibility)} \\ \mathcal{A}^\top(y) + Z = C,\ Z \in \mathcal{S}_n^+ & \qquad \textit{(Dual feasibility)} \\ ZX = 0 & \qquad \textit{(Complementary slackness)} \end{aligned}$$

Chapter 3

Review of known results

3.1 The stable set problem

In this section we report some of the fundamental results for the maximum stable set problem. Among the several equivalent formulations this problem admits, we are going to discuss its classic linear and quadratic formulation, and we are going to show how the Lovász theta function [52] can be derived as a relaxation of the latter. Then we review some of the valid linear inequalities for the stable set problem which will be useful for our work. At last, following the development of Galli and Letchford [21], we present a hierarchy of semidefinite relaxations starting from the theta function, further strengthened by linear constraints.

3.1.1 Standard formulations

Let $G = (V, E)$ a simple undirected graph and let us introduce a vector $x \in \{0,1\}^{|V|}$ of binary variables in which the i-th component of x states whether the node $i \in V$ belongs to the stable set or not. Then the exact standard formulation of the MSSP is given by the following 0-1 linear program:

$$\begin{aligned} \alpha(G) = \max \quad & e^\top x \\ \text{s.t.}\ & x_i + x_j \leq 1 \quad \forall\ \{i,j\} \in E \\ & x \in \{0,1\}^{|V|}\,. \end{aligned} \tag{3.1}$$

Inequalities (3.1) arise directly from the problem's definition, stating that for each edge $\{i,j\} \in E$ at most one of the nodes i and j can be taken into the solution. Another equivalent formulation of the MSSP can be obtained using quadratic constraints as follows

$$\begin{aligned}\alpha(G) = \max \quad & e^\top x \\ \text{s.t.} \ & x_i x_j = 0 \quad \forall \ \{i,j\} \in E \\ & x_i^2 = x_i \qquad \forall \ i \in V,\end{aligned} \tag{3.2}$$

where the first equation is equivalent to the edge inequalities in (3.1), while the last is easy to see that only 0-1 points satisfy such inequality. Indeed, optimizing over this quadratic formulation is hard as well, but it will be interesting to use this model to derive the Lovász theta function as a semidefinite relaxation of the latter, showing the capacity to express quadratic constraints as linear equations in terms of a certain “augmented” matrix Y.

The *stable set polytope* is usually denoted by STAB(G) [28] and it is defined as the convex hull in $\mathbb{R}_+^{|V|}$ of 0-1 solutions of (3.1)

$$\text{STAB}(G) = \text{conv}\left\{x \in \{0,1\}^{|V|} \mid (3.1) \text{ hold}\right\}.$$

Due to the hardness in optimizing a linear objective function over STAB(G), it is natural to consider its continuous relaxation, i.e.

$$\text{FRAC}(G) = \left\{x \in [0,1]^{|V|} \mid x_i + x_j \leq 1 \quad \forall \ \{i,j\} \in E\right\},$$

which defines the so-called *fractional stable set polytope.* If on the one hand, it is easy to optimize over FRAC(G), from the other hand this provides very weak upper bounds on $\alpha(G)$. As a matter of facts if G is connected, it can be proved that the vector $x^* = (\frac{1}{2}, \dots, \frac{1}{2})$ is always an optimal solution. Hence, a great effort has been devoted in research to improving this basic relaxation, as we will discuss in the following section.

3.1.2 Valid linear inequalities

Clique The study of the stable set polytope was initiated by Padberg [63]. Let $C \subseteq V$ inducing a clique in G. Then the inequality

$$\sum_{i \in C} x_i \leq 1, \tag{3.3}$$

is valid for STAB(G). He observed that clique inequalities dominates the edge inequalities and moreover he showed that (3.3) induces a facet of STAB(G) if and only if C induces a maximal clique in G. Graphs for which all the clique and non-negativity inequalities suffice to describe STAB(G) are called *perfect graphs.*

Odd hole and odd antihole Padberg also introduces the *odd cycle* inequalities. Let $H \subseteq V$ induce a simple cycle in G, with $|H|$ odd, then the inequality

$$\sum_{i \in H} x_i \leq \left\lfloor \frac{|H|}{2} \right\rfloor, \tag{3.4}$$

is valid for STAB(G). In the case of $|H| \geq 5$ and the cycle is chordless, the cycle is called hole and we refer to (3.4) as *odd hole* inequalities. When $V = H$, the odd hole inequality induces a facet of STAB(G). However, Padberg pointed out that the odd hole inequalities do not induce facets in general. To convert an odd hole inequality into a facet for a general graph it has to be *lifted* [80], i.e. appropriate coefficients must be computed for the other variables.

The complement graph of an odd hole is referred as an *odd antihole*. Let $A \subseteq V$ inducing an odd antihole in G, then Padberg showed that the *odd antihole* inequality

$$\sum_{i \in A} x_i \leq 2, \tag{3.5}$$

is valid for STAB(G). Again in general (3.5) are not facet-defining and must be lifted, unless $A = V$.

Web and antiweb Trotter [75] introduced two classes of valid linear inequalities. Let $p, q \in \mathbb{N}$ such that $p \geq 2q + 1$ and $q \geq 2$. Here, arithmetic modulo p is used. The $W(p, q)$ (shown in Figure 3.1a) is a graph with $V = \{1, \ldots, p\}$ and containing edges from node i to $\{i + q, \ldots, i - q\}$ for all $1 \leq i \leq p$. The $AW(p, q)$ (shown in Figure 3.1b) is the complement of the web $W(p, q)$, i.e. each node i is connected to $\{i - q + 1, \ldots, i + q - 1\}$. Therefore, every consecutive q nodes in an antiweb graph induces a clique.

Trotter [75] showed that the *web* inequality

$$\sum_{i \in W(p,q)} x_i \leq q, \tag{3.6}$$

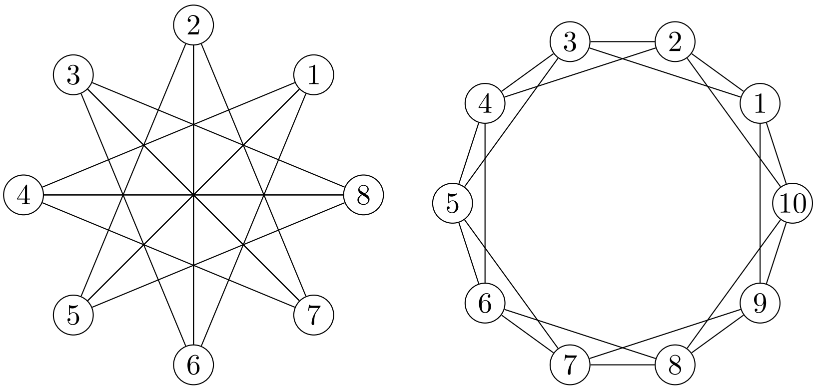
is valid for STAB(G). When $G = W(p, q)$, (3.6) is facet-defining if and only if p and q are relatively prime.

The *antiweb* inequality is defined as

$$\sum_{i \in AW(p,q)} x_i \leq \left\lfloor \frac{p}{q} \right\rfloor \tag{3.7}$$

and was proven to be valid for STAB(G) by Trotter. Laurent [45] showed that (3.7) is facet-defining for $G = AW(p, q)$ if and only if p is not a multiple of q. In general, for graphs G having web and/or antiweb as a node-induced subgraphs, both inequalities need to be lifted in order to become facet-defining.

We notice that web and antiweb graphs are quite general and they include many other remarkable graphs as special cases. $AW(p,2)$ with p odd (or equivalently $W(2q+1,q)$) are odd holes, whereas $AW(2q+1,q)$ (or equivalently $W(p,2)$, with p odd) are odd antiholes. Moreover, a clique on p vertices can be regarded as a "degenerate" web $W(p,1)$ or antiweb $AW(p,\lceil p/2\rceil)$.



(a) The $W(8,3)$ graph. **(b)** The $AW(10,3)$ graph.

Figure 3.1: An example of Antiweb and a Web graphs

Rank All the aforementioned inequalities belong to a more general class of valid inequalities for the stable set polytope, the so-called *rank* inequalities. Let $H \subseteq V$ any subset of nodes and let $G[H]$ be the subgraph induced by the nodes in H and let $\alpha(G[H])$ be its stability number, then the *rank* inequality corresponding to H is

$$\sum_{i\in H} x_i \leq \alpha(G[H]) \tag{3.8}$$

and clearly, it is valid for STAB(G). Finally, we remark that not all facet of STAB(G) are rank inequalities.

Odd wheel Let us consider $H \subset V$ inducing an odd hole and let $u \in V$ be a node adjacent to all others in H. The subgraph induced by the nodes $U = H \cup \{u\}$ is sometime referred as *odd wheel* (see [28], for example) and the corresponding inequality

$$\sum_{i\in H} x_i + \left\lfloor \frac{|H|}{2} \right\rfloor x_u \leq \left\lfloor \frac{|H|}{2} \right\rfloor \tag{3.9}$$

is called odd wheel inequality and is valid for STAB(G). If $V = U$, then (3.9) induces a facet of the stable set polytope. Such an inequality is one example of facet-inducing inequality which does not belong to the rank inequality class.

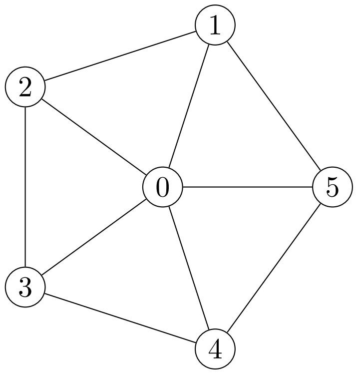


Figure 3.2: The 5-wheel graph.

Orthonormal representation Another class of non-rank constraints of a rather different character are the *orthonormal representation* inequalities, introduced by Grötschel, Lovász and Schrijver [27]. Let us associate with each vertex $i \in V$, a vector $v_i \in \mathbb{R}^n$, so that $||v_i|| = 1$ and non-adjacent vertices correspond to orthogonal vectors, i.e. $v_i \cdot v_j = 0$ for all $\{i, j\} \notin E$. Let $c \in \mathbb{R}^n$, an arbitrary vector with $||c|| = 1$. Then the constraints

$$\sum_{i \in V} (c^\top v_i)^2 x_i \leq 1, \qquad \forall\, i \in V, \tag{3.10}$$

are valid for the stable set polytope. These class of inequalities, along with the non-negativity defines TH(G) i.e., the *theta body* of G which, as we will discuss in the next section, is strictly related to the Lovásztheta function.

3.1.3 Semidefinite relaxations

In his seminal paper dated back in 1979, Lovász [52] introduced the well-known *theta function* $\theta(G)$ of a graph G which was proven to yield at the same time an upper bound on the stability number $\alpha(G)$ and a lower bound on the chromatic number $\chi(\bar{G})$ of the complement graph of G. Lovász presented several alternative derivations based on semidefinite programming. Subsequently to the publication of [52], SDPs were shown to be solvable in polynomial time, up to a fixed precision [76]. Then afterwards, different stronger relaxation have been presented through the addition of valid linear inequalities each of which based on two different formulations of the theta function, mainly. In order to report a complete picture about the hierarchy, we are going to report the two more common formulations (and the consequent strengthenings) following the survey of Galli and Letchford [21].

Theta function $\theta(G)$ The Lovász *theta function* of a graph $G = (V, E)$ can be derived starting from the quadratic formulation of the stable set problem (3.2) [28]. Let $|V| = n$, then for any feasible solution $x \in \{0,1\}^n$ we consider the augmented matrix (indexed from 0 up to n) defined as

$$Y := \begin{pmatrix} 1 \\ x \end{pmatrix} \begin{pmatrix} 1 \\ x \end{pmatrix}^\top = \begin{pmatrix} 1 & x^\top \\ x & xx^\top \end{pmatrix}.$$

We notice that Y is symmetric, positive semidefinite and has rank 1 (by Proposition 2.7). Moreover, since $x_i^2 = x_i$ we have that the first row of Y equals its diagonal. Now replacing the product $xx^\top$ with a matrix of variables $X \in \mathcal{S}_n$, with $(X)_{ij} = (X)_{ji} = x_{ij}$, an equivalent exact formulation for the MSSP is given by

$$\begin{aligned} \alpha(G) = \quad \max \quad & \operatorname{tr}(X) \\ \text{s.t.} \quad & Y_{00} = 1 \\ & x_{ij} = 0 \quad \{i,j\} \in E \\ & x_{ii} = x_{0i} \quad i \in V \\ & Y \in \mathcal{S}^+_{n+1}, \ \text{rank-1} \end{aligned} \quad \longrightarrow \quad \begin{aligned} \theta(G) = \quad \max \quad & \operatorname{tr}(X) \\ \text{s.t.} \quad & Y_{00} = 1 \\ & x_{ij} = 0 \quad \{i,j\} \in E \\ & x_{ii} = x_{0i} \quad i \in V \\ & Y \in \mathcal{S}^+_{n+1}, \end{aligned} \tag{th-SDP1}$$

where the right-hand side of the arrow is obtained by relaxing the rank-1 constraint, yielding a semidefinite relaxation of the stable set problem and whose optimum value is exactly the theta function $\theta(G)$.

The original formulation, introduced in [52] goes as follows. Given a feasible vector $x \in \{0,1\}^n$, define the matrix $Z = xx^T/(e^T x) \in [0,1]^{n \times n}$. By definition, Z is positive semidefinite and

$$\langle J, Z \rangle = (e^T x)^2/(e^T x) = e^T x.$$

Moreover, $\operatorname{diag}(Z) = x/(e^T x)$, which implies $\operatorname{tr}(Z) = 1$. Then $\theta(G)$ can again be computed by the SDP problem:

$$\begin{aligned} \theta(G) = \quad \max \quad & \langle J, Z \rangle \\ \text{s.t.} \quad & \operatorname{tr}(Z) = 1 \\ & Z_{ij} = 0 \quad \{i,j\} \in E \\ & Z \in \mathcal{S}^+_n. \end{aligned} \tag{th-SDP2}$$

Even if the optimal values of the two formulations are the same, the relationship between `th-SDP1` and `th-SDP2` is more involved than it might appear. `th-SDP2` has only $|E|+1$ constraints, whereas `th-SDP1` has $|V|+|E|$ linear constraints. On the other hand, the latter has a very nice property: its projection onto the subspace defined by the original (non-quadratic) x variables is usually referred as the *theta body* of G and it is denoted by $\mathrm{TH}(G)$. Such a set is always convex, but non-polyhedral in general as remarked in [53]. This is due to the fact that $\mathrm{TH}(G)$ is defined by orthogonality inequalities (3.10) (along with the non-negativity) which are infinitely many for a given graph. Remarkably, Grötschel et al. [28] proved that

$$\mathrm{STAB}(G) \subseteq \mathrm{TH}(G) \subseteq \mathrm{FRAC}(G),$$

furthermore he showed that $\mathrm{TH}(G)$ satisfies all clique constraints. This implies that when G is perfect, $\mathrm{TH}(G)$ is polyhedral and it describes the stable set polytope exactly. As a consequence, there exists a polynomial-time separation algorithm for a class of inequalities which includes all clique inequalities. This is so despite the fact that clique separation itself is strongly NP-hard. In practice, $\theta(G)$ often provides a strong bound to the stability number $\alpha(G)$, typically better than those obtained from standard linear relaxations [23, 24, 40, 83]. Another interesting feature of these SDPs concerns with their computational behaviour. In fact, they turn often out to be more tractable than general SDPs of comparable size by general SDP solvers. Moreover, very efficient tailored algorithms have been designed. The Lovász theta function can be computed efficiently for reasonable graph dimensions using augmented Lagrangian approaches as in Povh et al. [64] for example, or the regularization method of Malick et al. [56]. For these reasons, $\theta(G)$ achieves quite a good trade-off between strength of the upper bound to $\alpha(G)$ and computational burden.

Schrijver relaxation $\theta^+(G)$ According to Schrijver [69] non-negativity inequalities $x_{ij} \geq 0$ for all $\{i,j\} \notin E$, are added to `th-SDP1`, as these are not implied by the condition $Y \in \mathcal{S}^+_{n+1}$. Remember that $Y \succeq 0$ if and only if $b^\top Y b \geq 0$ for all vectors $b = (b_0, b_1, \ldots,\ b_n) \in \mathbb{R}^n$. Now letting $b_0 = \frac{1}{2}$, $b_i = b_j = -1$ for a pair $\{i,j\} \notin E$ and all the remaining elements to 0, $b^\top Y b \geq 0$ implies $x_{ij} \geq -\frac{1}{8}$. By the same arguments, $Z_{ij} \geq 0$ for all $\{i,j\} \notin E$ are added to `th-SDP2`. The resulting upper bounds are denoted by $\theta^+(G)$. The Schrijver's number slightly improves over $\theta(G)$ in practice, but already the addition of $Y \geq 0$ to the semidefinite program `th-SDP1` non-trivially increases the complexity of its resolution, such as for general purpose interior point methods for example. As a matter of facts, $Y \geq 0$ corresponds to $O(n^2)$ linear inequalities. On the other hand, augmented Lagrangian approaches overcome to this issue by handling lower (and upper) bounds on the matrix variable via a projection, see [14, 79, 82] for example.

Lovász-Schrijver relaxation This relaxation is obtained by applying the Lovász-Schrijver $N_+(\cdot)$ operator to $\mathrm{FRAC}(G)$. This is equivalent to adding the following linear inequalities to `th-SDP1`:

$$
\begin{aligned}
x_{ij} &\geq 0 && \{i,j\} \notin E, && (3.11)\\
x_{ik} + x_{jk} &\leq x_k && \{i,j\} \in E,\ k \in V,\ k \neq i,j, && (3.12)\\
x_i + x_j + x_k &\leq 1 + x_{ik} + x_{jk} && \{i,j\} \in E,\ k \in V,\ k \neq i,j. && (3.13)
\end{aligned}
$$

Lovász and Schrijver [53] showed that $N_+(\mathrm{FRAC}(G))$ satisfies all clique, odd hole, odd antihole and odd wheel inequalities. Giandomenico and Letchford [22] showed that, in fact, it satisfies all web inequalities. Thus, SDP provides a polynomial-time separation algorithm for a class of inequalities which includes all web and odd wheel inequalities (and therefore all clique, odd hole and odd antihole inequalities). A characterization of graphs for which $N_+(\mathrm{FRAC}(G))$ completely describes the stable set polytope has been presented in Bianchi et al. in [5] and more recently in [6].

Computational results obtained by optimizing over $M_+(\mathrm{FRAC}(G))$ have been given for example by Balas et al. [3] (using lift-and-project cutting plane methods) and Burer and Vandenbussche [10], where a specialised augmented Lagrangian method is devised to deal with these SDPs. These upper bounds are often noticeably better than $\theta^+(G)$ but at the expense of very large running times. In practice, this relaxation turns out to be hardly computationally accessible for graphs of medium/large size.

Gruber-Rendl relaxation The Gruber and Rendl relaxation, introduced in [29], is obtained by adding further inequalities to LS formulation, namely

$$
\begin{aligned}
x_{ik} + x_{jk} &\leq x_k + x_{ij} && \forall \text{ stable } \{i,j,k\}, && (3.14)\\
x_i + x_j + x_k &\leq 1 + x_{ij} + x_{ik} + x_{jk} && \forall \text{ stable } \{i,j,k\}. && (3.15)
\end{aligned}
$$

They use an interior-point SDP algorithm and incorporate the triangle inequalities by Lagrangian relaxation. Near-optimal multipliers are then computed via the bundle method. The upper bounds obtained are very good, although again at the expense of large running times.

Dukanovic-Rendl relaxation Dukanovic and Rendl [19] instead proposed to add the following set of inequalities to `th-SDP2`:

$$
\begin{aligned}
Z_{ij} &\geq 0 && \{i,j\} \notin E, && (3.16)\\
Z_{ik} + Z_{jk} &\leq Z_{kk} && \{i,j\} \in E, k \in V, k \neq i,j, && (3.17)\\
Z_{ik} + Z_{jk} &\leq Z_{kk} + Z_{ij} && \forall \text{ stable } \{i,j,k\}. && (3.18)
\end{aligned}
$$

Computational results documented in [19] show that the upper bounds provided by this relaxation are strong. Here computational results are reported as well, but running times are demanding on reasonable graph dimensions.

We finally mention that a quite strong bound $\alpha(G)$ (indeed presented in the context of the clique number) has been achieved by Locatelli [50] by adding non-valid inequalities. The resulting method turns out to be one of the most powerful among those currently available.

The general picture provided by these relaxations is that the additional computational cost to be paid in order to improve the theta bound is often quite significant. Indeed, the insertion of linear inequalities in the basic Lovász model yield much harder SDPs and does require specialised methods. In the next chapter we introduce a new relaxation with the purpose of bridging such a gap.

3.2 Graph Coloring Problem

In this section we are going to report some basic concepts about the graph coloring problem. As it has been done for the MSSP, we are going to present the classical integer linear formulations accordingly to the surveys in [36, 55]. Hence, we review the main semidefinite relaxations known for this problem starting again from the Lovász theta function and further improved through the addition of valid linear inequalities.

3.2.1 Standard formulations

Given a graph $G = (V, E)$ with $|V| = n$, the natural ILP formulation can be obtained by the introduction of a vector $x \in \{0,1\}^{n\times n}$ and a vector $w \in \{0,1\}^n$, where the general entry x_{vi}, $v \in V$, $i \in [n]$ assumes value 1 if and only if the node v is assigned to color i, while $w_i = 1$ if and only if the color i is used (i.e. it has been assigned to at least one vertex). Then the GCP can be formulated by

$$\begin{aligned} \chi(G) = \quad \min \quad & \textstyle\sum_{i=1}^n w_i \\ \text{s.t.} \quad & \textstyle\sum_{i=1}^n x_{vi} = 1 \qquad \forall\, v \in V \\ & x_{ui} + x_{vi} \le w_i \quad \forall\, \{i,j\} \in E,\ i \in [n] \\ & x \in \{0,1\}^{n\times n}, \\ & w \in \{0,1\}^n . \end{aligned} \tag{GCP-ASS}$$

This formulation is usually referred as *assignment formulation* and it has the advantage to be intuitive and can be easily adapted to variants of the GCP. The equality constraints state that each vertex is assigned to exactly one color. The inequality constraint has a twofold meaning: from the one hand it enforces each adjacent vertices to be assigned to different colors, while on the other hand ensures that if some node receives color i, then w_i must be set to 1. In this way, the objective function correctly minimizes the number of used colors. Here we presented this model using $|V|$ as the maximum number of colors needed, but in many cases (unless the graph is a clique, for example) this number can be lowered (and hence reducing the number of variables and constraints) to some tighter upper bound, such as the result of a heuristic. Let H be such a number, then this model employs $H|V| + H = O(|V|^2)$ variables and $|V| + H|E| = O(|V||E|)$ inequalities. Malaguti and Toth [55] pointed out that beyond the fact that colors are indistinguishable and then there exists many symmetric solutions, the LP relaxation is extremely weak: it is easy to verify that $x_{v1} = x_{v2} = \frac{1}{2}$ and $x_{vj} = 0$, $w_1 = w_2 = 1$ and $w_j = 0$ for $\forall\ v \in V$ and $j = 3, \ldots, H$ is always a feasible solution of value 2, for any graph G.

Since any k-coloring in G defines a partition of the nodes V into k stable sets, Mehrotra and Trick [57] proposed the so-called *set covering formulation*. Let $\mathcal{S}$ be the collection of all stable sets in G and $S(i) \subseteq \mathcal{S}$ be the subset of stable sets including vertex i. Moreover, let x_s a binary variable for each stable set $s \in \mathcal{S}$ with value 1 if and only if the stable set s belongs to the partition and 0 otherwise, then an alternative formulation of GCP is given by

$$\begin{aligned}
\chi(G) = \quad \min \quad & \textstyle\sum_{s \in \mathcal{S}} x_s \\
\text{s.t.} \quad & \textstyle\sum_{s \in S(i)} x_s \geq 1 \qquad \forall\ v \in V \\
& x \in \{0,1\}^{|\mathcal{S}|}\,.
\end{aligned} \tag{GCP-COV}$$

Here, the objective function calls for the minimum number of stable sets employed to partition (i.e. to color) the vertices in V. The inequality constraints enforce that each vertex belongs at least to one stable set. One drawback of this formulation resides in the number of variables, which is exponential in the number of nodes in G. In order to face this issue Mehrotra and Trick proposed a branch-and-price algorithm, starting with a compact number of variables and then adding new ones using column generation methodologies. Along this approach, Gualandi and Malucelli [30] proposed a method employing branch-and-price enhanced by constraint programming to compute exact solutions for the graph coloring problem. The linear relaxation of the set covering formulation is quite interesting, indeed relaxing the integrality constraint with $x \in [0,1]^{|\mathcal{S}|}$ we allow each node to be assigned to a set of fractions of colors. As a matter of fact it is referred to the optimal value of such a relaxation as the *fractional chromatic number* of G, denoted as $\chi^f(G)$ [43]:

$$\begin{aligned} \chi^f(G) = \quad \min \quad & \textstyle\sum_{s \in \mathcal{S}} x_s \\ \text{s.t.} \quad & \textstyle\sum_{s \in S(i)} x_s \geq 1 \qquad \forall\, v \in V \\ & x \in [0,1]^{|\mathcal{S}|}. \end{aligned} \tag{FGCP}$$

The dual of FGCP corresponds to the fractional clique problem, a relaxation of the clique problem whose optimal value is denoted by $\omega^f(G)$. By strong duality in LP, the following inequality holds:

$$\omega(G) \leq \omega^f(G) = \chi^f(G) \leq \chi(G).$$

We also mention the alternative linear formulation for the graph coloring based on the definition of a partial ordering proposed by Jabrayilov and Mutzel [36] and further strengthened in [37] recently. In Chapter 5 we investigate a new SDP relaxation based on the LP representative formulation proposed by Campelo et al. [12].

3.2.2 Semidefinite relaxations

Similarly to upper bounds on the stable set, the computation of lower bounds for $\chi(G)$ has been intensively investigated with the purpose of achieving a good trade-off between quality of the bound and efficiency. Again, lower bounds from linear relaxations [58, 61] are cheap to compute but may be rather weak, while lower bounds from semidefinite programming relaxations are typically stronger but also harder to handle in practice. The SDP approaches are based on the fundamental result of Lovász [52] which states that

$$\alpha(G) \leq \theta(G) \leq \chi(\bar{G}).$$

Hence, computing the parameter $\theta(G)$ already provides a lower bound on the chromatic number on the complement graph $\bar{G}$. In literature many equivalent formulations have been presented, we refer to Laurent and Rendl [48] for a detailed comparison. Given a graph $G = (V, E)$, let $\bar{G} = (V, \bar{E})$ be its complement graph, then an alternative to formulation th-SDP2 was proposed by Meurdesoif [59] and it goes as follows:

$$\begin{aligned} \theta(G) = \quad \min \quad & t \\ \text{s.t.} \quad & X_{ii} = t - 1 \qquad v \in V \\ & X_{ij} = -1 \qquad \{i,j\} \in \bar{E}, \\ & X \in \mathcal{S}_n^+. \end{aligned} \tag{th-SDP3}$$

Again, the Lovász theta number provides a good trade-off between quality of the bound and computational burden. On the other hand, Lovász [52] also proved that $\theta(G) \leq \chi^f(\bar{G})$. Thus, the gap $\chi(\bar{G}) - \theta(G)$ tends to increase as $\chi^f(\bar{G})$ gets closer to $\alpha(G)$. In general, such a gap may become arbitrarily large as in the case of the triangle free ($\omega(G) = 2$) Mycielski graphs [43].
In order to improve the lower bound towards the chromatic number, several attempts have been made with adding linear inequalities to the Lovász SDP relaxation towards $\chi(\bar{G})$ [19, 59, 74].

The $\theta'(G)$ parameter In a similar fashion to the Schrijver's number, Szegedy [74] proposed to replace $Z_{ij} = 0$ to $Z_{ij} \leq 0$ for $\{i, j\} \in E$ to `th-SDP2` as a sharpening for the chromatic number of $\bar{G}$. Independently, Meurdesoif [59] added $X_{ij} \geq -1$ to `th-SDP3`. The resulting lower bound, denoted as $\theta'(G)$ is the same for the two formulations

$$\begin{aligned} \theta'(G) = \quad \max \quad & \langle J, Z\rangle \\ \text{s.t.} \quad & \mathrm{tr}(Z) = 1 \\ & Z_{ij} \leq 0 \quad \{i,j\} \in E \\ & Z \in \mathcal{S}_n^+. \end{aligned} \qquad \begin{aligned} \theta'(G) = \quad \min \quad & t \\ \text{s.t.} \quad & X_{ii} = t - 1 \qquad v \in V \\ & X_{ij} = -1 \qquad \{i,j\} \in \bar{E}, \\ & X_{ij} \geq -1 \qquad \{i,j\} \in E, \\ & X \in \mathcal{S}_n^+. \end{aligned}$$

The resulting lower bound improves over the initial $\theta(G)$. Meurdesoif also presented graphs for which the inequality $\theta(G) \leq \theta'(G)$ is strict.

Triangle inequalities In his paper, Meurdesoif [59] proposed a further strengthening by the introduction of the so-called *triangle inequalities*. Those inequalities are a standard way to improve relaxations in which variables correspond to some Boolean variables. Indeed, the exact formulation for the graph coloring behind `th-SDP3`, asks the entries X_{ij} to assume values in $\{t-1, -1\}$, whether i and j are assigned to the same color or not. Thus, he proposed to add the following family of nontrivial inequalities:

$$X_{ij} + X_{jk} - X_{ik} \leq t - 1 \qquad \{i,j\}, \{j,k\} \in E$$

The resulting relaxations typically require tailored algorithms to be handled. In particular, a computational progress has been documented in [19] by using different models for sparse and dense graphs and by exploiting symmetry. However, these studies seldom presented lower bounds which lie above the fractional chromatic number. This threshold has been crossed in [33] by a powerful operator and computational results have been presented in [32]. The latter have been again obtained by a sophisticated algorithm embedding an advanced symmetry-exploiting mechanism. Moreover, Gaar and Rendl [20] recently presented computational results on the so-called exact subgraph hierarchy for the GCP where SDP bounds above $\chi^f(\bar{G})$ have been presented.

3.3 The Matrix-Cut Operator

In this section we describe the general framework introduced by Lovász and Schrijver [53] to derive valid inequalities (i.e. cutting planes), called matrix cuts, for 0-1 vectors in polyhedra. Such a construction "lifts" a 0-1 problem K in n variables to $O(n^2)$ variables, and then projecting it back to the n-space so that cuts, i.e. tighter inequalities still valid for all 0-1 solutions, are introduced. Such Lift-and-Project methods owe their success by the following principle: the convex hull of 0-1 points of K whose number of facets are exponential, might be represented as a projection of another convex set Q "living" in a higher (but still polynomial) dimensional space, since the projection of Q onto the K-space might have more facets than Q in its lifted space. Beside Lovász and Schrijver, several related works have been presented by Sherali and Adams [71], Balas et al. [2] and Lasserre [44]. A detailed comparison of these methods have been presented in Laurent [46]. A common feature of these methods is the construction of a hierarchy of relaxations, which typically involve several levels, each of which gets increasingly tighter to the exact solution, but the computational effort grows accordingly. Our interest will be in the first level of the $N_+(\cdot)$ hierarchy.

3.3.1 The construction of matrix-cuts

Let Q_n be the 0-1 cube in $\mathbb{R}^n$, that is $Q_n = [0,1]^n$. If the dimension is obvious from the context, we denote the 0-1 cube by Q. Let P be a polytope defined by

$$P = \{x \in Q \mid Ax \leq b\}, \tag{3.19}$$

where $Ax \leq b$ is the system of m linear inequalities $a_i^\top x \leq b_i, \forall\, i \in [m]$, defining P. We assume, for the sake of convenience, that the inequalities $0 \leq x \leq 1$ are included in the system. We are interested in determining (or approximating) P^I, i.e. the convex hull of all 0-1 vectors in P.

In what is following, it will be convenient to work with homogeneous systems of inequalities, i.e., with convex cones rather than polytopes. In order to accomplish that, we can obtain a convex cone from P by embedding the n-dimensional space in $\mathbb{R}^{n+1}$ within the hyperplane $x_0 = 1$. Now, let us consider the following set

$$\bar{P} := \left\{ \begin{pmatrix} 1 \\ x \end{pmatrix} \in \mathbb{R}^{n+1} \mid u_i^\top \begin{pmatrix} 1 \\ x \end{pmatrix} \geq 0, \ \forall\, i \in [m] \right\}, \tag{3.20}$$

where $u_i := \begin{pmatrix} b_i \\ -a_i \end{pmatrix}$. Clearly, $\bar{P}$ is a convex cone in $\mathbb{R}^{n+1}$ and we have that $P = \left\{ x \mid \begin{pmatrix} 1 \\ x \end{pmatrix} \in \bar{P} \right\}$. We denote by $\bar{P}^I$ the cone spanned by all 0-1 vectors in $\bar{P}$ (i.e. the convex hull of its 0-1 vectors) and let $\bar{P}^*$ be its polar cone, i.e. the cone spanned by vectors u_i, for $i \in [m]$. The same process can be done with Q_n, considering it as the set spanned by all 0-1 vectors $x \in \mathbb{R}^{n+1}$, with $x_0 = 1$, hence

$$\bar{Q} = \left\{ \begin{pmatrix} 1 \\ x \end{pmatrix} \in \mathbb{R}^{n+1} \mid e_i^\top \begin{pmatrix} 1 \\ x \end{pmatrix} \geq 0, \ (e_0 - e_i)^\top \begin{pmatrix} 1 \\ x \end{pmatrix} \geq 0, \ \forall\, i \in [n] \right\}. \tag{3.21}$$

Let $\bar{Q}^*$ be the polar cone of $\bar{Q}$, denote by e_i, $i = 0, \dots, n$ the ith unit vector, and let $f_i := e_0 - e_i$, then $\bar{Q}^*$ is spanned by all vectors e_i and f_i.

Lovász and Schrijver noticed that if we consider any pair of valid inequalities for $\bar{P}$ and $\bar{Q}$ respectively (i.e. vectors belonging to their polar cones) and we multiply them together, we get a (quadratic) inequality which is still valid for every point in $\bar{P}$. To exemplify, let us consider $a_i^\top x \leq b_i$ and $x_j \leq 1$. Clearly they represent two inequalities defining P and Q for some $i \in [m]$ and $j \in [n]$, then all points in $\bar{P}$ and $\bar{Q}$ must satisfy $b_i - a_i^\top x \geq 0$ and $1 - x_j \geq 0$ equivalently and since they are both non-negative also the product $(b - a_i^\top x)(1 - x_j) \geq 0$ will be valid for the cone $\bar{P}$ (and hence for P).

Let $u \in \bar{P}^*$ and $v \in \bar{Q}^*$, by the discussion above we remark that the following holds

$$0 \leq \left(u^\top \begin{pmatrix} 1 \\ x \end{pmatrix} \right) \left(\begin{pmatrix} 1 \\ x \end{pmatrix}^\top v \right) = \left\langle uv^\top, \begin{pmatrix} 1 \\ x \end{pmatrix} \begin{pmatrix} 1 \\ x \end{pmatrix}^\top \right\rangle, \tag{3.22}$$

for any $x \in P$. Inequalities generated as in (3.22) are quadratic. In order to linearize them we need the following

Remarks 3.1. 1. *Given a feasible 0-1 vector $x \in P^I \subseteq P$, the matrix*

$$Y = \begin{pmatrix} 1 \\ x \end{pmatrix} \begin{pmatrix} 1 \\ x \end{pmatrix}^\top = \begin{pmatrix} 1 & x^\top \\ x & xx^\top \end{pmatrix},$$

satisfies inequalities (3.22)*;*

2. *Since $x_i x_j = x_j x_i$, $Y \in \mathcal{S}_{n+1}$ and by Proposition 2.7 $Y \in \mathcal{S}_{n+1}^+$ and has rank 1;*

3. *All 0-1 vectors $x \in P^I$ (and no other point in P) also satisfy:*

$$x_i^2 = x_i, \text{ for all } i \in [n],$$

or equivalently, $\operatorname{diag}(Y) = Ye_0$. *Therefore, by the substitution in* (3.22) *of x_i^2 with x_i, the inequality remains valid for 0-1 points in P^I, but not necessarily for other points in P. Hence we call* (3.22) *a* matrix-cut *for P.*

By the remarks above, we can now replace the product $xx^\top$ in Y with a new symmetric matrix $X \in \mathcal{S}_n$, where $(X)_{ij} = x_{ij}$ and $\operatorname{diag}(X) = x$. Then the inequality (3.22) becomes

$$\left\langle uv^\top, \begin{pmatrix} 1 & x^\top \\ x & X \end{pmatrix} \right\rangle \geq 0, \tag{3.23}$$

which is linear in the entries of Y. After this discussion we are ready to describe the steps needed to apply the Lovász and Schrijver Lift-and-Project operator to the polytope P.

3.3.2 Lift-and-Project operator

Let P be the polytope defined by (3.19). Then the operator involves three steps:

Step 1. Lift Generate the set of nonlinear inequalities

$$(b_i - a_i^\top x)x_j \geq 0, \tag{3.24}$$

$$(b_i - a_i^\top x)(1 - x_j) \geq 0, \quad \text{for } i \in [m], j \in [n]. \tag{3.25}$$

Step 2. Linearize Define the augmented matrix

$$Y := \begin{pmatrix} 1 & x^\top \\ x & X \end{pmatrix},$$

with $X \in \mathcal{S}_n$, $(X)_{ij} = x_{ij}$ and $\operatorname{diag}(X) = x$. Then equations (3.24) and (3.25) become

$$\left\langle u_i e_j^\top, Y \right\rangle \geq 0, \tag{3.26}$$

$$\left\langle u_i (e_0 - e_j)^\top, Y \right\rangle \geq 0, \quad \text{for } i \in [m], j \in [n]. \tag{3.27}$$

Now define the following cone

$$M_+(P) := \left\{ \begin{pmatrix} 1 & x^\top \\ x & X \end{pmatrix} \in \mathcal{S}^+_{n+1} \;\middle|\; (3.26) \text{ and } (3.27) \text{ hold, with } \operatorname{diag}(X) = x \right\}.$$

Step 3. Project The projection of $M_+(P)$ onto the original space is defined as

$$N_+(P) := \left\{ \operatorname{diag}(X) \mid \begin{pmatrix} 1 & x^\top \\ x & X \end{pmatrix} \in M_+(P) \right\}.$$

3.3.3 Properties of the matrix cut operator

The focus of this section is to outline the main properties of $M_+(\cdot)$ and $N_+(\cdot)$ operators just introduced. For the sake of compactness, in what is following we are going to omit some of the proofs. We refer the reader to the original paper [53] for a complete treatment.

The first, crucial question is whether the result of the operator $N_+(P)$ is actually a relaxation of P^I stronger than P, or not. Indeed Lovász and Schrijver proved the following

Lemma 3.2. *$P^I \subseteq N_+(P) \subseteq P$.*

Proof. $P^I \subseteq N_+(P)$. By Remarks 3.1.1-2, the matrix

$$Y = \begin{pmatrix} 1 & x^\top \\ x & xx^\top \end{pmatrix},$$

is symmetric, positive semidefinite and satisfies (3.26) and (3.27), for any 0-1 point $x \in P^I$.

$N_+(P) \subseteq P$. For any $i \in [m]$ and any $j \in [n]$, we have

$$b_i - a_i^\top x = (b_i - a_i^\top x)x_j + (b_i - a_i^\top x)(1 - x_j),$$

then inequalities defining P can be obtained as a convex combination of inequalities defining $N_+(P)$. □

The previous lemma gives us the certificate of $N_+(P)$ to be a relaxation of P and the inclusion is strict in general. In order to provide a better understanding of the construction of the $N_+(\cdot)$ operator, Lovász and Schrijver observed the following geometrical property:

Lemma 3.3. *For every polytope $P \subseteq Q$ and every $i \in [n]$,*

$$N_+(P) \subseteq \operatorname{conv}((P \cap \{x \mid x_i = 0\}) \cup (P \cap \{x \mid x_i = 1\})).$$

Let us point out the consequences of this lemma: clearly, $\{x \mid x_i = 0\}$ and $\{x \mid x_i = 1\}$ are facet of Q, and all facets of Q are determined in this way. Hence, if P does not intersect some facet of Q, then $N_+(P)$ is contained in the opposite facet. Furthermore, if P does not intersect some pair of opposite facets of Q, then $N_+(P) = \emptyset$.

Another crucial point to address that will be important for our purpose, is to answer the following:

Question. *Given an inequality $c^\top x \leq d$ (or $d - c^\top x \geq 0$), is such an inequality valid for $N_+(P)$?*

In order to answer, Lovász and Schrijver provided two main theoretical tools which will be useful to report now. The first tool is given by the next Lemma, defining an explicit description of the polar cone $M_+(P)^*$, containing valid constraints for $M_+(P)$.

Lemma 3.4. *An inequality $d - c^\top x \geq 0$ is valid for $M_+(P)$ if and only if*

$$\begin{aligned}\begin{pmatrix} d \\ -c \end{pmatrix} e_0^\top = &\sum_{i,j} \alpha_{i,j} u_i e_j^\top + \\ &\sum_{i,j} \beta_{i,j} u_i (e_0 - e_j^\top) + \\ &\sum_j \lambda_j e_j (e_0 - e_j^\top) + \boldsymbol{A} + \boldsymbol{B},\end{aligned} \tag{3.28}$$

where $\alpha_{i,j},\ \beta_{i,j} \geq 0, \lambda_j \in \mathbb{R}$, for $i \in [m]$, $j \in [n]$, $\boldsymbol{A}$ is a skew-symmetric matrix and $\boldsymbol{B}$ is a symmetric positive semidefinite matrix. Then $d - c^\top x \geq 0$ is called an N_+-cut (or matrix-cut) for P.

The polar cone $M_+(P)^*$ is precisely the set of matrices having the same form as the right-hand side of (3.28): the first two summations can only yield convex combination of inequalities (3.26) and (3.27), the last summation achieves the same purpose as replacing x_i^2 by x_i, while matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are related to the symmetry and positive semidefiniteness of matrix Y, respectively.

The other tool to certificate the validity of an inequality for $N_+(P)$ is a sufficient condition stated by the following

Lemma 3.5. *[53, Lemma 1.5] Let $c \geq 0$. If $c^\top x \leq d$ is valid for $(P \cap \{x \mid x_i = 1\})$ for all $c_i > 0$, then $c^\top x \leq d$ is valid for $N_+(P)$.*

Notice that the Lemma assume $c \geq 0$. This can be done without loss of generality as shown in the original paper [53] (by flipping coordinate i, if $c_i < 0$). During our application of the $N_+(\cdot)$ to the stable set problem, we will use this tool, since its use can be easily verified in the graph

(a) $P \subseteq Q$ and its convex hull of 0-1 points P^I **(b)** $N_+(P)$

Figure 3.3: The application of $N_+(\cdot)$ to P

G on which the problem is defined (by the removal of some node i). As a matter of fact, by the employment of this lemma, Lovász and Schrijver proved that $N_+(FRAC(G))$ satisfies clique, odd-hole, odd-antihole and odd-wheel inequalities. In the negative direction, Bianchi et al. [5] observed that not every valid inequality of $N_+(FRAC(G))$ satisfies these conditions.

3.3.4 An example of the $N_+(\cdot)$ operator

In this section we report an example of $N_+(\cdot)$ construction, in order to clarify what we discussed until now. The following example is reported in Dash's PhD dissertation [16]. Let us consider the two-dimensional unit cube $Q = [0,1]^2$, and let $P \subseteq Q$ be the polytope (as shown in Figure 3.3a) defined by the inequalities

$$\begin{aligned} 1.5 - x_1 - x_2 &\geq 0, \\ 0.5 + x_1 - x_2 &\geq 0, \\ x_1 \geq 0,\ 1 - x_1 &\geq 0, \\ x_2 \geq 0,\ 1 - x_2 &\geq 0, \end{aligned} \tag{3.29}$$

in order to derive $N_+(P)$, we proceed by applying the steps described in Section 3.3.2.

Step 1. Lift Since $1.5 - x_1 - x_2$ and x_1 must be non-negative for every point in P, the quadratic inequality $(1.5 - x_1 - x_2)x_1 \geq 0$ will be valid for P. We repeat this process for each of $2mn$ possible combinations ($m = 6,\ n = 2$ in our example). Then the following (non-trivial) inequalities are valid for P:

$$
\begin{aligned}
(1.5 - x_1 - x_2)x_1 &\geq 0, \\
(0.5 + x_1 - x_2)x_2 &\geq 0, \\
(1.5 - x_1 - x_2)x_2 &\geq 0, \\
(0.5 + x_1 - x_2)(1 - x_1) &\geq 0,
\end{aligned}
\tag{3.30}
$$

Step 2. Linearize All 0-1 points in P satisfy $x_i^2 = x_i$. Then by replacing x_1^2 by x_1, in the first equation in (3.30), $1.5 - x_1^2 - x_1 x_2 \geq 0$, we obtain $0.5x_1 - x_1 x_2 \geq 0$. Now we replace the product $x_1 x_2 = x_2 x_1$ with a new variable x_{12}. Repeating this process for all equations in (3.30) and collecting all variables in the augmented matrix Y, we can infer that $M_+(P)$ is defined by the following system:

$$
\begin{aligned}
0.5x_1 - x_{12} &\geq 0, \\
-0.5x_2 + x_{12} &\geq 0, \\
0.5x_2 - x_{12} &\geq 0, \\
0.5 - 0.5x_1 - x_2 + x_{12} &\geq 0, \\
Y = \begin{pmatrix} 1 & x_1 & x_2 \\ x_1 & x_1 & x_{12} \\ x_2 & x_{12} & x_2 \end{pmatrix} &\in \mathcal{S}_3^+.
\end{aligned}
\tag{3.31}
$$

Step 3. Project The projection of (3.31) onto $\mathbb{R}^2$ yields the inequalities

$$
\begin{aligned}
x_1 - x_2 &\geq 0, \\
1 - x_1 - x_2 &\geq 0,
\end{aligned}
\tag{3.32}
$$

which define $N_+(P)$, as shown in Figure 3.3b. Then (3.32) are N_+-cuts, valid for P^I but not for P. We notice that in this special case, $N_+(P)$ is polyhedral but in general this is not the case, since the condition $Y \succeq 0$ is equivalent to an infinite number of linear inequalities. Furthermore, we remark that it might be the case in which not all inequalities generated by the lifting in Step 1 are needed for the description of $M_+(P)$ and then $N_+(P)$, and the identification of such "redundant" part of the lifting is not straightforward.

Chapter 4

Application of $N_+(\cdot)$ operator to $\mathrm{NOD}(G)$

The Lovász theta function $\theta(G)$ has been a game changer since its introduction: it provides an exact polynomial-time algorithm for solving the MSSP on a special subclass of graphs, while for general graphs it yields stronger upper bounds than linear ones achieving a hard-to-beat trade-off between computational effort and strength of the bound. As shown in Section 3.1.3 many attempt of strengthen $\theta(G)$ bounds have been proposed but although their success, the general picture is that the additional computational cost to be paid in order to yield a significant improvement is non-trivial, resulting in SDPs harder to solve with general purpose algorithms.

In last section we reviewed the Lovász and Schrijver matrix-cut operator which allows to strengthen any linear relaxation of a general 0-1 linear problem into a semidefinite one in an elegant fashion based on cones' properties. The application of such operator to the fractional stable set polytope proposed by Lovász and Schrijver, namely $N_+(\mathrm{FRAC}(G))$, showed the potential of this procedure to yield strong bounds on the stability number, as remarked by the results in Burer and Vandenbussche [10] for example. Although the strength of these bounds is alluring, dealing with $N_+(\cdot)$ SDPs tends to be computationally challenging and the employment of tailored algorithms is often required.

This chapter deals with the following study: we investigate a new semidefinite relaxation for the MSSP obtained by the application of Lovász and Schrijver Lift-and-Project operator to an alternative compact formulation for the stable set problem. At first, theoretical aspects are addressed in order to provide a comparison with the well known $N_+(\mathrm{FRAC}(G))$, following the development of the original paper [53]. Then extensive computational experiments are reported, discussing on how to deal with the computational burden using general purpose SDP methods.

4.1 Nodal inequalities

As remarked in Section 3.3, when the lifting operator $M_+(\cdot)$ is applied, $2nm$ valid linear inequalities are generated (n and m being the number of variables and the number of constraints defining the polytope on which the operator is applied, respectively). For example, FRAC(G) requires $O(n^2)$ inequalities, then $M_+(\text{FRAC}(G))$ size would be of $O(n^3)$. Clearly, this implies that the starting linear formulation plays a crucial role on the tractability of the resulting SDP. An alternative formulation for the stable set problem has been proposed by Murray and Church [60], based on the so called *nodal inequalities*. Before introducing their formulation, some notation is needed. Let $G = (V, E)$ a simple undirected graph, then for each $i \in V$ we denote with $\Gamma(i) \subset V$ the neighborhood set of i, i.e. $\Gamma(i) = \{j \in V \mid \{i, j\} \in E\}$. Given any $S \subseteq V$, we denote with $G[S]$ the subgraph induced by the nodes in S and by $r(G[S])$ its stability number. Then Murray and Church's formulation reads as follows:

$$
\begin{aligned}
\alpha(G) = \quad \max \quad & e^\top x \\
\text{s.t.} \quad & \textstyle\sum_{j\in\Gamma(i)} x_j + r(G[\Gamma(i)])x_i \le r(G[\Gamma(i)]) \quad i \in V \qquad (4.1)\\
& x \in \{0,1\}^{|V|},
\end{aligned}
$$

where equations (4.1) have the following meaning: in any stable set S, if $i \in S$ then no other nodes $j \in \Gamma(i)$ in its neighborhood can also be in S; otherwise if $i \notin S$, then at most $r(G[\Gamma(i)])$ among adjacent to i can be in S. Remarkably, (4.1) are exactly $|V| = \Theta(|V|)$ in number, one for each node; while the standard edge formulation has $|E| = O(|V|^2)$ inequalities. We denote with

$$
\text{NOD}(G) = \Big\{ x \in [0,1]^{|V|} \mid (4.1) \text{ hold } \Big\}
$$

the polytope associated with its continuous relaxation. Then we report the following

Remark 4.1. *The computation of $r(G[\Gamma(i)])$ coefficients needed to write* (4.1) *has been deeply investigated by Letchford, Rossi and Smriglio [49]. Even if it may involve large CPU times, they show how to deal with this process in practice for the graph's sizes of interest, proposing a decomposition technique to enhance the computation.*

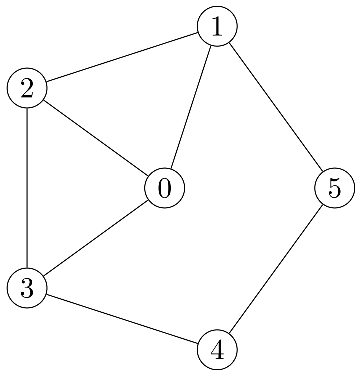


Figure 4.1: An almost-wheel graph G_{LT}

Remark 4.2. NOD(G) *does not satisfy the edge inequalities in general. As shown in Figure 4.1, let us consider the graph* G_{LT} *as reported in [5], i.e. the 5-hole with an additional* 0*-node and let it adjacent to three consecutive nodes in the hole, say* $\{1, 2, 3\}$ *for example. Let* $\bar{x} = \left(\frac{2}{3}, \frac{2}{3}, 0, 0, 0, 0\right)$*, it's easy to verify that* $\bar{x} \in \text{NOD}(G_{LT})$*, but on the other hand the edge inequality* $x_0 + x_1 \leq 1$ *is clearly violated by* $\bar{x}$*.*

Remark 4.3. *When G is an odd wheel graph,* NOD(G) *trivially satisfies the odd wheel inequality. Indeed equation* (4.1) *corresponding to the hub of the wheel, is equivalent to* (3.9)*.*

Murray and Church [60] proposed a more general family of formulations based on a combination of (4.1) and a collection of clique inequalities, but here we consider only the "pure" nodal formulation.

4.2 Lifting the nodal polytope

In this section we describe the matrix-cuts generated by the application of $M_+(\cdot)$ to NOD(G). Again let $n = |V|$, then we denote with $Q = [0, 1]^n$ the 0-1 cube of the right dimension and recall the definition of the augmented matrix

$$Y := \begin{pmatrix} 1 & x^\top \\ x & X \end{pmatrix},$$

with $X \in \mathcal{S}_n$, $(X)_{ij} = (X)_{ji} = x_{ij}$ and $\text{diag}(X) = x$. For the sake of compactness, we write r_i in place of $r(G[\Gamma(i)])$. To apply Step 1 of the procedure described in Section 3.3.2, we need to multiply (4.1) by $x_k \geq 0$ and $1 - x_k \geq 0$, $k \in [n]$. Hence, we report all possible classes of matrix-cuts (i.e. inequalities) generated by an exhaustive case analysis. For the sake of simplicity, we implicitly perform the linearization of the inequalities (i.e. by the substitutions $x_i = x_i^2$ and $x_{ij} = x_i x_j$ for all i and j).

Multiplying by $x_k \geq 0$ We need to develop the following product:

$$x_k(r_i - \sum_{j\in\Gamma(i)} x_j - r_i x_i) \geq 0,$$

for every $i, k \in [n]$. Here, one out of three possible cases can occur:

1. $(i = k)$, then we obtain
$$-\sum_{j\in\Gamma(i)} x_{ij} \geq 0; \tag{4.2}$$

2. $(i \neq k,\ k \in \Gamma(i))$, then we obtain
$$(r_i - 1)x_k - \sum_{j\in\Gamma(i)\setminus\{k\}} x_{jk} \geq 0; \tag{4.3}$$

3. $(i \neq k,\ k \notin \Gamma(i))$, then we obtain
$$r_i x_k - \sum_{j\in\Gamma(i)} x_{jk} - r_i x_{ik} \geq 0. \tag{4.4}$$

Multiplying by $1 - x_k \geq 0$ In this case, we need to develop the following product:

$$(1 - x_k)(r_i - \sum_{j\in\Gamma(i)} x_j - r_i x_i) \geq 0,$$

for every $i, k \in [n]$. By the same argument as before, the three possible cases are:

1. $(i = k)$, then we obtain
$$r_i - \sum_{j\in\Gamma(i)} (x_j - x_{ij}) - r_i x_i \geq 0; \tag{4.5}$$

2. $(i \neq k,\ k \in \Gamma(i))$, then we obtain
$$r_i - \sum_{j\in\Gamma(i)\setminus\{k\}} (x_j - x_{jk}) - r_i x_i - r_i x_k + r_i x_{ik} \geq 0; \tag{4.6}$$

3. $(i \neq k,\ k \notin \Gamma(i))$, then we obtain
$$r_i - \sum_{j\in\Gamma(i)} (x_j - x_{jk}) - r_i x_i - r_i x_k + r_i x_{ik} \geq 0. \tag{4.7}$$

As remarked in Lovász and Schrijver [53], since $\mathrm{NOD}(G)$ is contained in Q, every $Y \in M_+(\mathrm{NOD}(G))$ satisfies, in addition

$$x_{ij} \geq 0, \tag{4.8}$$

$$x_{ij} \leq x_i, \tag{4.9}$$

$$x_i + x_j - x_{ij} \leq 1, \quad \text{for all } i, j \in [n] \tag{4.10}$$

$$x_i \leq 1, \quad \text{for all } i \in [n]. \tag{4.11}$$

This set of (non-trivial) constraints is the result of the lifting corresponding to the bound constraints $0 \leq x \leq 1$ in $\mathrm{NOD}(G)$ (which are assumed to be in the description of the polytope). The bound constraint (4.11) is implied by the condition $Y \succeq 0$ since by Proposition 2.8, $x_i \geq 0$ and the determinant of all submatrices of Y of the form

$$\begin{pmatrix} 1 & x_i \\ x_i & x_i \end{pmatrix},$$

for $i \in [n]$ must be non-negative, implying the bounds $0 \leq x_i \leq 1$. Moreover, the determinant of all submatrices of Y of the form

$$\begin{pmatrix} x_i & x_{ij} \\ x_{ij} & x_j \end{pmatrix},$$

implies the bound $x_{ij} \leq 1$ for all i and j.

Hence, we can now define the convex set

$$M_+(\mathrm{NOD}(G)) = \left\{ \begin{pmatrix} 1 & x^\top \\ x & X \end{pmatrix} \in \mathcal{S}^+_{n+1} \mid \mathrm{diag}(X) = x,\ (4.2)\text{–}(4.10) \text{ hold} \right\},$$

along with the corresponding projection onto the original space

$$N_+(\mathrm{NOD}(G)) := \left\{ \mathrm{diag}(X) \mid \begin{pmatrix} 1 & x^\top \\ x & X \end{pmatrix} \in M_+(\mathrm{NOD}(G)) \right\}.$$

4.3 On the strength of $N_+(\mathrm{NOD}(G))$

In this section we investigate the relationships between $N_+(\mathrm{NOD}(G))$ and some of the previous relaxations. In order to do that from a theoretical viewpoint, we are going to prove which class of valid inequalities reviewed in Section 3.1.2, are satisfied by this new semidefinite relaxation. We recall that Grötschel et al. [28] proved that $\mathrm{TH}(G)$ satisfies all clique constraints, while Lovász and Schrijver showed that the application of the operator to the edge formulation $N_+(\text{FRAC}(G)) \subseteq \text{TH}(G)$ and furthermore it satisfies all odd wheel inequalities, odd hole and odd antihole. Giandomenico et al. [23] showed that the more general class of web inequalities (which includes all odd holes and antiholes as special cases) is satisfied. A characterization of graphs for which $N_+(\text{FRAC}(G))$ completely describes the stable set polytope has been presented in Bianchi et al. in [5] and more recently in [6].

By Remark 4.2, $\text{NOD}(G)$ does not imply edge inequalities in general. Hence the question is whether this is still the case after the lifting or not. With the next lemma we are going to prove a stronger result, that is $N_+(\text{NOD}(G))$ satisfies all clique inequalities (and hence all edge inequalities, by dominance).

Lemma 4.4. $N_+(\text{NOD}(G))$ *satisfies all clique inequalities.*

Proof. Let $G = (V, E)$ be any simple undirected graph and let $C \subseteq V$ inducing a maximal clique in G. In order to apply Lemma 3.5 to the clique inequality

$$\sum_{i \in C} x_i \leq 1,$$

we need to show that

$$\sum_{j \in C \setminus \{i\}} x_j \leq 0,$$

is valid for $\text{NOD}(G) \cap \{x \mid x_i = 1\}$ for all $i \in C$.
Let i be any node in C, then the nodal inequality corresponding to i is

$$\sum_{j \in \Gamma(i)} x_j + r(G[\Gamma(i)])x_i \leq r(G[\Gamma(i)]),$$

then $x_i = 1$, along with x being non-negative imply $x_j = 0$ for all $j \in \Gamma(i)$ but clearly $C \subseteq \Gamma(i)$, hence the statement holds. □

Next, we consider the orthonormal representation constraints. In order to prove this result we will use the containment relation given by the following

Lemma 4.5. $N_+(\text{NOD}(G)) \subseteq \text{TH}(G)$.

Proof. We can show that the inequalities $x_{ij} = 0,\ \forall \{i, j\} \in E$ are satisfied by all solution matrices of $M_+(\text{NOD}(G))$. Notice that any edge $(i, j) \in E$ belongs to exactly two neighbor sets, namely $\Gamma(i)$ and $\Gamma(j)$. Then, if we sum up all of the inequalities (4.2) we obtain $\sum_{\{i,j\} \in E} x_{ij} \leq 0$. Since $x_{ij} \geq 0$

by (4.8) for any $i, j \in V$ (and, in particular, for $\{i, j\} \in E$), this implies $x_{ij} = 0$ for $\{i, j\} \in E$. This shows that $M_+(\mathrm{NOD}(G))$ is contained in the feasible region of `th-SDP1`. The result follows from projection onto the x space. □

Since all points in $N_+(\mathrm{NOD}(G))$ are in $\mathrm{TH}(G)$, we can state the following

Corollary 4.6. $N_+(\mathrm{NOD}(G))$ *satisfies all orthonormal representation constraints.*

Furthermore, we remark that $M_+(\mathrm{NOD}(G))$ is also contained in the feasible region of the Schrijver's number, since the non-negativity constraints $x_{ij} \geq 0$ for $\{i, j\} \notin E$ are included in the description of $M_+(\mathrm{NOD}(G))$ as well. Hence, the optimum over the latter is at least as good as $\theta^+(G)$, by the previous Lemma.

Now we are going to address a more complicated, yet interesting issue, that is in which way $N_+(\mathrm{NOD}(G))$ and $N_+(\mathrm{FRAC}(G))$ are related each other. As we have just seen, both applications of the Lift-and-Project operator are a strengthening of the Lovász theta function and hence they both satisfy all clique and orthonormal representation constraints. Since by Lemma 3.2, all linear constraints in the description of $\mathrm{NOD}(G)$ are obtainable as a convex combination of $M_+(\mathrm{NOD}(G))$ (by construction) then we can state the following

Corollary 4.7. $N_+(\mathrm{NOD}(G))$ *satisfies all odd wheel inequalities.*

The same holds for $N_+(\mathrm{FRAC}(G))$, as showed by Lovász and Schrijver [53]. Therefore a straightforward question is what happens when we look at web and antiweb inequalities. Let us consider the web graph $W(8, 3)$ (as shown in Figure 3.1a), by the result provided by Giandomenico et al. [23] $N_+(\mathrm{FRAC}(W(8, 3)))$ implies the facet-defining web inequality $\sum_{i=1}^{8} x_i \leq 3$, hence if we optimize over it we get an integer solution with value 3. On the contrary, the optimum yield by $N_+(\mathrm{NOD}(W(8, 3)))$ is $(0.42675, \ldots, 0.42675)$ with value 3.414. This suggests two important aspects:

- $N_+(\mathrm{NOD}(G))$ does not satisfies the class of web inequalities in general;
- there exists a point for which $\bar{x} \in N_+(\mathrm{NOD}(W(8, 3)))$ but $\bar{x} \notin N_+(\mathrm{FRAC}(W(8, 3)))$.

On the other hand, if we consider the antiweb graph $AW(10, 3)$ (as shown in Figure 3.1b) we are able to identify a complementary result. Indeed, the optimal value obtained over $N_+(\mathrm{NOD}(AW(10, 3)))$ correspond to $\alpha(AW(10, 3)) = 3$, while optimizing over $N_+(\mathrm{FRAC}(AW(10, 3)))$ yields the fractional solution $(0.31055, \ldots, 0, 31055)$ with value 3.1055. Again the following considerations can be inferred:

- $N_+(\mathrm{FRAC}(G))$ does not satisfies the class of antiweb inequalities in general;
- $N_+(\mathrm{NOD}(AW(10, 3)))$ implies the antiweb inequality $\sum_{i=1}^{10} x_i \leq 3$;

- there exists a point for which $\bar{x} \in N_+(\mathrm{FRAC}(AW(10,3)))$ but $\bar{x} \notin N_+(\mathrm{NOD}(AW(10,3)))$.

By these points, we can immediately generalize that there is no containment relationship between $N_+(\mathrm{NOD}(G))$ and $N_+(\mathrm{FRAC}(G))$. More interestingly, it turns out that the satisfaction of the antiweb inequalities can be generalized as well. Indeed, with the next theorem we are going to prove that $N_+(\mathrm{NOD}(G))$ satisfies almost all inequalities belonging to such a class.

Theorem 4.8. *$N_+(\mathrm{NOD}(G))$ satisfies all $AW(p,q)$ inequalities such that $p \mod q \neq q-1$.*

Proof. Let $G = AW(p,q)$. Here we denote the stability number with α, in place of $\alpha(G)$ for short. We are going to prove the statement by duality i.e. that the matrix $we_0^\top$ belongs to the dual cone $M_+(\mathrm{NOD}(G))^*$, where $w := (\alpha, -1, \ldots, -1) \in \mathbb{R}^{p+1}$. For this purpose, it is sufficient to show that the inequality $\alpha - \sum_{i \in V} x_i \geq 0$ can be obtained as a convex combination of inequalities defining $M_+(\mathrm{NOD}(G))$. Hence, the result will follow by the projection onto the x-space.

First, we consider the case in which $\alpha = \left\lfloor \frac{p}{q} \right\rfloor \geq 4$. Recall that $Y \succeq 0$ if and only if $b^T Y b \geq 0$ for all $b \in \mathbb{R}^{p+1}$. Thus, the positive semidefiniteness condition on Y implies the following explicit family of linear inequalities, the so-called *psd inequalities*:

$$b_0^2 + \sum_{i \in V}(2b_0 b_i + b_i^2)x_i + 2 \sum_{1 \leq i \leq j \leq p} b_i b_j x_{ij} \geq 0, \tag{4.12}$$

where $b = (b_0, b_1, \ldots, b_p)^T$ is an arbitrary real vector. We construct the *psd inequality* with $b_0 = \sqrt{\alpha}$ and $b_i = -\frac{\sqrt{\alpha}}{\alpha}$ for $i = 1, \ldots, p$ and by Lemma 4.5, since $x_{ij} = 0 \ \forall \ \{i,j\} \in E$ is implied by $N_+(\mathrm{NOD}(G))$, we can remove them from the inequality:

$$\alpha + \left(\frac{1}{\alpha} - 2\right) \sum_{i \in V} x_i + \frac{2}{\alpha} \sum_{\{i,j\} \notin E} x_{ij} \geq 0. \tag{4.13}$$

Here, arithmetic modulo p is used. For each $\bar{i} = 1, \ldots, p$ select the inequalities (4.3), for $i = \bar{i}$ and for $k = \bar{i} + q - 1$ and its symmetric $k' = \bar{i} - q + 1$ with coefficient 1. Then again, for each $\bar{k} = 1, \ldots, p$ consider the subset of nodes $h = \{\bar{k} + 2(q-1) + 1, \ldots, \bar{k} - 2(q-1) - 1\}$. Starting from $i = \bar{k} + 2(q-1) + 1$ and $i' = \bar{k} - 2(q-1) - 1$, select inequalities (4.4) with coefficient 1 for $k = \bar{k}$ and i, i'. Now iterate the selection increasing (resp. decreasing) i (resp. i') by q, until i and i' would cross one another. Then consider the subset $\bar{h} = \{i+1, \ldots, i'-1\}$ of h. If $|\bar{h}| < q$ no other inequalities (4.4) are needed. Otherwise, one out of the two following cases can occur:

- if $|\bar{h}|$ is odd let $i_{\bar{h}}$ be the element in position $\left\lceil \frac{|\bar{h}|}{2} \right\rceil$ (i.e. the element in the middle of $\bar{h}$) and select the inequality (4.4) for $k = \bar{k}$ and $i_{\bar{h}}$ with coefficient 1;

- if $|\bar{h}|$ is even let $i_{\bar{h}}$ and $i'_{\bar{h}}$ be the elements in position $\frac{|\bar{h}|}{2}$ and $\frac{|\bar{h}|}{2}+1$ respectively (i.e. the two elements in the middle of $\bar{h}$) and select the inequalities (4.4) for $k=\bar{k}$ and $i_{\bar{h}}$, $i'_{\bar{h}}$ with coefficient $\frac{1}{2}$ both.

Summing up all the selected inequalities yields

$$(2\alpha-2)\sum_{i\in V} x_i - 4\sum_{\{i,j\}\notin E} x_{ij} \geq 0. \tag{4.14}$$

We remark that in some cases in order to get the inequality (4.14), summing inequalities $x_{ij} \geq 0$ for some $\{i,j\} \notin E$ and for some $c \in \mathbb{R}_+$ along with those above mentioned, may be needed.
Now, multiply (4.14) by $\frac{1}{2\alpha} > 0$ and sum it to (4.13) in order to get

$$\alpha - \sum_{i\in V} x_i \geq 0.$$

Now consider the case in which $\alpha = 2$. By the same arguments as before, we construct the psd inequality (4.13)

$$\alpha + \left(\frac{1}{\alpha} - 2\right)\sum_{i\in V} x_i + \frac{2}{\alpha}\sum_{\{i,j\}\notin E} x_{ij} \geq 0.$$

For each $\bar{i} = 1,\ldots,p$ select the inequalities (4.3), for $i=\bar{i}$ and for $k=\bar{i}+q-1$ and its symmetric $k' = \bar{i}-q+1$ with coefficient 1. Summing up all the selected inequalities yields

$$2\sum_{i\in V} x_i - 4\sum_{\{i,j\}\notin E} x_{ij} \geq 0. \tag{4.15}$$

Now, multiply (4.15) by $\frac{1}{2\alpha} > 0$ and sum it to (4.13) in order to get the antiweb inequality.

At last, consider the case in which $\alpha = 3$. Again, we construct the psd inequality (4.13)

$$\alpha + \left(\frac{1}{\alpha} - 2\right)\sum_{i\in V} x_i + \frac{2}{\alpha}\sum_{\{i,j\}\notin E} x_{ij} \geq 0.$$

For each $\bar{i} = 1,\ldots,p$ select the inequalities (4.3), for $i=\bar{i}$ and for $k=\bar{i}+q-1$ and its symmetric $k' = \bar{i}-q+1$ with coefficient 1. Summing up all the selected inequalities yields

$$2\sum_{i\in V} x_i - 2\sum_{\{i,j\}\notin E} x_{ij} \geq 0. \tag{4.16}$$

Now, multiply (4.16) by $\frac{1}{\alpha} > 0$ and sum it to (4.13) in order to get the antiweb inequality. $\square$

On the negative side, when $p \bmod q = q-1$ we are unable to apply the steps reported in

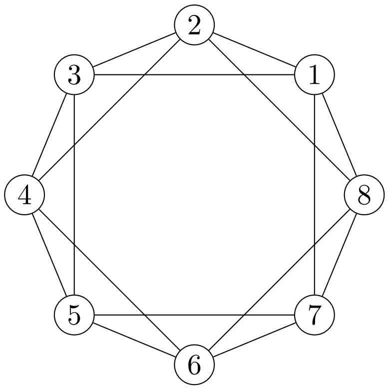


Figure 4.2: The $AW(8,3)$ graph.

the previous proof. To exemplify, consider the $AW(8,3)$ as shown in Figure 4.2. Optimizing over $N_+(\mathrm{NOD}(AW(8,3)))$, the optimal solution is $(0.2929,\ldots,0.2929)$ yielding an optimal value of 2.343, hence the inequality $\sum_{i\in V} x_i \le 2$ is clearly violated.

Despite this fact, a large subset of antiweb inequalities are satisfied by our application of the Lovász and Schrijver operator. Similarly to the web graphs, due to their generality we can extend this result to odd holes and odd antiholes. Indeed, $AW(p,2)$ with p odd are odd holes, whereas $AW(2q+1,q)$ are odd antiholes. On the negative side, since $p \bmod q = 1 = q-1$ for $q = 2$ and for any p odd, we have the following

Corollary 4.9. *Odd hole inequalities are not implied by* $N_+(\mathrm{NOD}(G))$.

On the contrary, since for any $q > 2$, $2q+1 \bmod q = 1 \neq q-1$ we have to following result

Corollary 4.10. *Odd antihole inequalities are implied by* $N_+(\mathrm{NOD}(G))$ *for* $|H| > 5$.

In conclusion, we have shown the existence of an "anti-symmetric" behaviour between the application of the same Lift-and-Project operator to two different linear formulations for the MSSP, namely the edge and nodal formulations:

- on the one hand, $N_+(\mathrm{FRAC}(G))$ satisfies all clique, orthonormal representation, odd wheel, odd hole, odd antihole and web inequalities;
- on the other hand, $N_+(\mathrm{NOD}(G))$ satisfies all clique, orthonormal representation, odd wheel, odd antihole and almost all antiweb inequalities.

In the next section we are going to investigate the behaviour of these two relaxations in practice. We are going to face the computational challenge arising by the SDPs yield from the applications of the Lovász and Schrijver Lift-and-Project operator. As we will see, the difficulty of facing $N_+(\mathrm{FRAC}(G))$ reported in Burer and Vandenbussche [10], will be confirmed in our computational experiments. Despite being more compact and easier to keep in memory, the computational burden needed to solve $N_+(\mathrm{NOD}(G))$ with semidefinite algorithms is non-trivial as well and one need to address it properly. On the positive side, we will show how to effectively deal with these hard formulations deploying general alternating direction methods of multipliers for large-scale semidefinite programs.

4.4 Numerical results

In this section we discuss our computational study. Here the main goal is to compare these different SDP relaxations in terms of their strength and computational tractability with general purpose algorithms for SDPs. The comparison we propose is not limited to $N_+(\mathrm{NOD}(G))$ and Lovász and Schrijver's $N_+(\mathrm{FRAC}(G))$: in order to describe a clearer picture we extend it also to other relaxations of the hierarchy reported in Section 3.1.3, namely Gruber-Rendl and Dukanovic-Rendl (denoted by GR and DR for simplicity in what is following). In general, these SDP relaxations raise a computational challenge due to the number of constraints which increases quickly w.r.t. the size of the graph. Hence the selection of an appropriate general purpose SDP algorithm is an issue to address: despite interior-point methods being well-established tools for small and medium sized SDPs providing a good accuracy in a reasonable time, they become impractical for large scales semidefinite programs (i.e. hundreds of thousands of constraints) due to memory requirements. On the other hand, alternating direction methods of multipliers (ADMMs) represent a competitive alternative for solving large scale SDPs. In particular, for this computational study we have made use of `SDPNAL+` [82] a state-of-the-art `MATLAB` software awarded with the Beale-Orchard-Hays Prize in 2018, implementing an ADMM combined with a semismooth Newton-Conjugate Gradient method.

Despite `SDPNAL+` being able to handle such large instances, preliminary experiments in which we fed to the solver the whole formulations were unsatisfactory since the computational burden to solve them was too demanding. As a matter of facts, the addition of such a number of inequalities to the Lovász theta function's basic formulations (both `th-SDP1` and `th-SDP2`) makes the convergence of these SDPs harder to reach in a reasonable time, in general. Hence, in order to mitigate the computational burden we have investigated the employment of a cutting-plane scheme, as shown in Algorithm 1: taking advantage of the fact that relaxations $N_+(\mathrm{NOD}(G))$, $N_+(\mathrm{FRAC}(G))$, GR and DR can be seen as a strengthening of the Schrijver's number $\theta^+(G)$, instead of solving the whole formulations one can start by solving the Schrijver's relaxation and then adding the most violated constraints (if any). This process is then repeated until either no more violated constraints are found or the objective value is not improving substantially. Such a cutting-plane method was originally presented in Kelley's seminal paper [42], where at each iteration, the separation problem is solved by selecting the most violated inequalities by the current optimum X^* in the linear system

$\mathcal{A}(X^*) \leq b$.

For this numerical experiment we employed an `Intel Xeon CPU E5-2698 v4` running at `2.20GHz`, equipped with `256GB` of `RAM`, under `Linux` (`Ubuntu 16.04.7`). The solver `SDPNAL+` has been used with its default settings, that is an optimality tolerance set at 10^{-6}. At each iteration of the cutting-plane scheme we select at most 1000 violated hyperplanes and we stop it if the improvement of the objective value is less than $\varepsilon = 10^{-1}$, imposing one hour as a time limit for each instance. The code employed to formulate the SDPs can be found at `https://github.com/batt95/SDP_MSSP_GCP`.

Algorithm 1 Cutting-plane scheme for MSSP SDP relaxations

Choose $c \in \mathbb{N}$, $\varepsilon \in \mathbb{R}_+$
Let $\mathrm{R} = \{X \in \mathcal{S}_n^+ \ : \ \mathcal{A}(X) \leq b\}$ be the relaxation to solve
Let $\mathrm{TH}^+ \supseteq \mathrm{R}$ be the Schrijver's relaxation
$\delta \leftarrow +\infty$
$C \leftarrow \{\}$
Compute $p^* = \mathrm{opt}(\mathrm{TH}^+)$ and let X^* be its optimal solution
while X^* violates constraints in R **or** $\delta > \varepsilon$ **do**
    Let V be the set of c most violated constraints by X^* in $\mathcal{A}(X^*) \leq b$
    $C \leftarrow C \cup V$
    Compute $p^{new} = \mathrm{opt}(\mathrm{TH}^+ \cap C)$ and let X^{new} be its optimal solution
    $\delta \leftarrow |p^{new} - p^*|$, $X^* \leftarrow X^{new}$, $p^* \leftarrow p^{new}$
end while

4.4.1 Instances

The numerical experiments are based on the following collections of graphs.

- Erdös–Rényi random graphs: $G(n,p)$ where $n \in \mathbb{N}$ is the number of nodes in the graph and $p \in [0,1]$ is the edge probability. In particular, for this experiment we have taken instances proposed in Letchford, Rossi and Smriglio [49] that is, instances with

  $$n \in \{150, 175, 200, 225, 250, 275, 300\}, \quad p \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}.$$

  Moreover for each combination of n and p, 5 different instances have been created for a total of 315 graphs. These graphs can be downloaded from `http://optimization.disim.univaq.it/stableset/` in the edge-list format.

- graphs from the Second DIMACS Implementation Challenge (see [39]). These graphs form the standard benchmark for max-clique algorithms. We complemented the graphs to convert the max-clique instances into MSSP instances. We selected graphs among `brock`, `C`, `DSJC`, `hamming`, `johnson`, `keller`, `MANN`, `p-hat` and `sanr` families. The final collection includes 38 graphs.

- graphs from the Sloane Independent Set Challenge [72] and in particular 1zc-instances are known to be hard instances for the MSSP. As a matter of facts, for most of them the optimal solutions are still unknown. Here we selected only two instances, namely 1zc512 and 1zc1024.

4.4.2 Comparison among relaxations

As remarked before, our main focus is to measure the strength of the bounds obtained by $N_+(\mathrm{NOD}(G))$ compared to other the SDP relaxations belonging to the hierarchy described in Section 3.1.3, in particular $N_+(\mathrm{FRAC}(G))$, GR and DR. For the first two relaxations and $N_+(\mathrm{NOD}(G))$ we are going to start the cutting-plane algorithm by solving the formulation based on `th-SDP1` of the Schrijver's number $\theta^+(G)$, while the equivalent based on `th-SDP2` will be used for the latter.

As a customary, given a graph G and a relaxation R we measure the so-called *percentage gap*, that is the following quantity:

$$\mathrm{Gap}(\mathrm{R}, G) := \frac{\mathrm{R}(G) - \alpha(G)}{\alpha(G)},$$

where $\mathrm{R}(G)$ is the optimal value obtained by solving the corresponding SDP and $\alpha(G)$ is the stability number of G (if $\alpha(G)$ is unknown, then the best lower bound is used in place). When no confusion arises we denote such a quantity by Gap, for short.

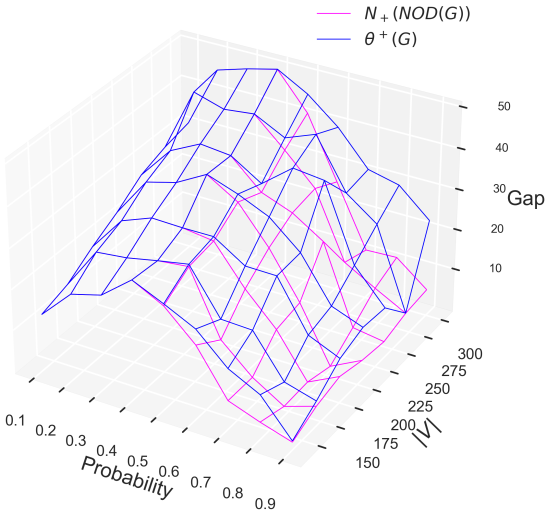
In addition we report, the total number of violated cuts (denoted by Cuts) that have been added during the cutting-plane scheme along with the total CPU resolution time in seconds (denoted by CPU-time).

Random instances

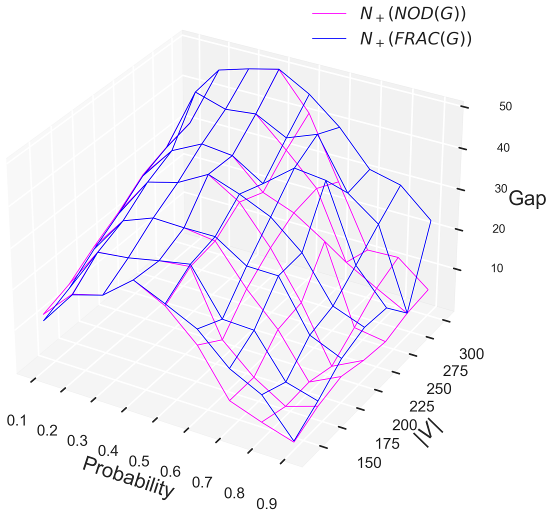
In Table 4.1 results on random instances are shown. In addition to $N_+(\mathrm{NOD}(G))$, $N_+(\mathrm{FRAC}(G))$, GR and DR we reported the Gap obtained by the linear relaxation $\mathrm{NOD}(G)$ and by the starting value $\theta^+(G)$ (along with the CPU-time needed to solve the formulation based on `th-SDP1`), as a reference. For every pair of values n and p, each entry reports the arithmetic mean computed over the 5 different instances. Moreover, in Figure 4.3 three-dimensional plots show the comparison among gaps yield by $N_+(\mathrm{NOD}(G))$ with $\theta^+(G)$, $N_+(\mathrm{FRAC}(G))$, GR and DR respectively, varying for each density and number of nodes in the graphs. Here, we were able to formulate and solve all 315 instances for each relaxation.

In particular, $N_+(\mathrm{NOD}(G))$ improves the upper bound over $\theta^+(G)$ in 205 cases out 315, while $N_+(\mathrm{FRAC}(G))$, GR and DR improve over the latter in 110, 111 and 17 out of 315 instances, respectively. Furthermore, if we look at the linear formulation $\mathrm{NOD}(G)$ we can notice that its semidefinite counterpart is significantly better on all instances, confirming the strength of the Lovász and Schrijver's Lift-and-Project operator. Moreover, the gap obtained by $N_+(\mathrm{NOD}(G))$ outperforms the $N_+(\mathrm{FRAC}(G))$ gap in 193 instances out of 315.

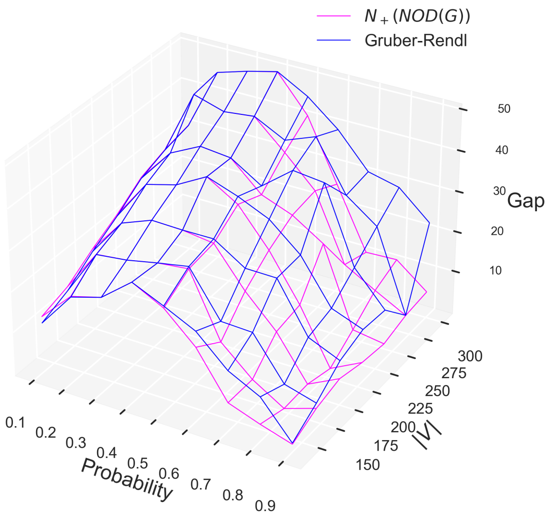
The quality of the bounds are also confirmed by the numbers of added violated cuts which point out a clear message: in $N_+(\mathrm{FRAC}(G))$ and Gruber-Rendl the major number of violated cuts can be found on graphs with $p = 0.1$ and it decreases as soon as the density goes up, while when the density reaches 50% almost none can be found. Although this behaviour can be measured also when the size of the graph increases, the number of cuts and hence the improvement of the bound is less prominent. DR relaxation has a similar, yet less marked, behaviour: violated cuts can be found on very sparse graphs, but as soon as the size of the graph increases, the number of cuts goes down.



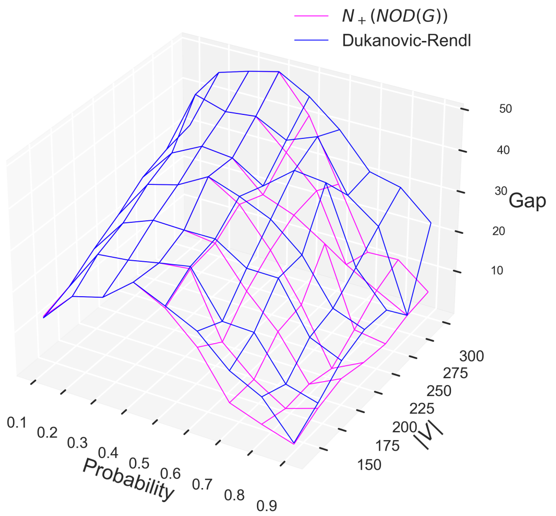
(a) $N_+(\mathrm{NOD}(G))$ and $\theta^+(G)$



(b) $N_+(\mathrm{NOD}(G))$ and $N_+(\mathrm{FRAC}(G))$



(c) $N_+(\mathrm{NOD}(G))$ and Gruber-Rendl



(d) $N_+(\mathrm{NOD}(G))$ and Dukanovic-Rendl

Figure 4.3: Percentage gap of semidefinite relaxations on Erdös–Rényi $G(n, p)$ random graphs

A complementary result can be identified on $N_+(\mathrm{NOD}(G))$, where violated cuts are found as soon as the density hits $p = 0.4$, reaching the peak between 0.7 and 0.8. Moreover, the number of cuts increases accordingly to the number of nodes. Such consistency in the number of cuts is reflected on the quality of the bound as well, managing to almost close the gap on very dense graphs (80-90%). This behaviour is also remarked in Figure 4.3, where the slope of the magenta lines ($N_+(\mathrm{NOD}(G))$) increases at $p = 0.4$ w.r.t. the blue lines ($\theta(G)$, $N_+(\mathrm{FRAC}(G))$, GR and DR, resp.). On the contrary, the latter are slightly under the magenta lines on very sparse graphs.

Moreover, we point out that the maximum improvement of GR is achieved on $n = 150$, $p = 0.1$ and is about of 1.5 percentage point, which is remarkably for such a sparsity. On the other hand, where $N_+(\mathrm{NOD}(G))$ shows an improvement, its Gap is significantly lower not only over $\theta(G)$, but also over $\mathrm{NOD}(G)$ which achieves a good Gap on these instances already.

About the computational tractability, these results confirmed our perspective. Since violated cuts have been identified in $N_+(\mathrm{NOD}(G))$, $N_+(\mathrm{FRAC}(G))$, GR and DR in 213, 130, 131 and 17 cases out of 315 (resp.) and when found the total amount is a small portion w.r.t. the total number of constraints for those formulations, the cutting-plane scheme helps to mitigate the computational burden compared to the time needed by the whole formulations to converge. Even though there are cases where the computation of multiple SDPs are needed to solve only one relaxation, the trade-off is positive for this approach. As a matter of fact, `SDPNAL+` managed to reach the accuracy required for each of these subproblems in a reasonable amount of time, leading to only one additional order of magnitude of overhead in average over the resolution of $\theta^+(G)$ (except for $n = 225$ and $p = 0.8$, where $N_+(\mathrm{NOD}(G))$ needed on average about 200 seconds to converge but it almost closes the gap). At last, we remark that the resolution time of DR may be different to the time to compute $\theta^+(G)$, even without the addition of violated cuts. This is due to the fact that the time reported for $\theta^+(G)$ is related to formulation `th-SDP1`, while Dukanovic and Rendl's relaxation is based on `th-SDP2` and hence the behaviour of the solver may vary.

DIMACS instances

In Table 4.2 results on DIMACS instances are reported. In this case we listed the Gap obtained by relaxations $\theta^+(G)$, $N_+(\mathrm{NOD}(G))$, $N_+(\mathrm{FRAC}(G))$, GR and DR, the number of total violated Cuts added during the cutting-plane and the CPU-time needed to reach the optimality. While for $N_+(\mathrm{NOD}(G))$ we were always able to formulate the SDPs, we failed to build the $N_+(\mathrm{FRAC}(G))$, GR and DR's models due to memory requirements in 12 out of 40 cases.

In 15 out of 40 instance $N_+(\mathrm{NOD}(G))$ significantly improves over $\theta^+(G)$. In particular, on p_hat300-1, p_hat500-1 and p_hat700-1 it almost closes the gap, while on brock200_2, DSJC125.5, DSJC500-5 and sanr400_0.5 it outperforms the other SDP relaxations. Surprisingly, while on smaller instances brock200 and brock400, $N_+(\mathrm{NOD}(G))$ does not change the Gap significantly, an improvement of up to 2% has been achieved on large instances brock800. On the 28 instances where we could solve $N_+(\mathrm{FRAC}(G))$ and GR relaxations, they show a similar behaviour, improving over $\theta^+(G)$ in 11 cases and in particular they significantly outperforms the other relaxations on C125-9, DSJC125.1, MANN_a9, p_hat300-3, p_hat500-2 and sanr200_0.9. DR's relaxation is slightly weaker than $N_+(\mathrm{FRAC}(G))$, indeed we were able to identify violated cuts in 5 out of 28 instances while on the remaining cases, it equals $\theta^+(G)$'s Gap.

Similarly to random graphs, also on DIMACS instances the cutting-plane scheme matched our perspectives. Violated cuts have been found in 27 out of 40 cases for $N_+(\mathrm{NOD}(G))$, in 16 out of 28 cases for $N_+(\mathrm{FRAC}(G))$ and GR, in 5 out of 28 instances for DR. Again, the number of separated cuts is far less to the total number of constraints in these formulations, yielding a computational overhead of at most two order of magnitude only w.r.t. the time needed to solve the starting formulation of $\theta^+(G)$. Exceptions can be found in p_hat700-1 for $N_+(\mathrm{NOD}(G))$, where the bound improves significantly and in MANN_a27 for GR.

		NOD(G)	$\theta^+(G)$		$N_+(\text{NOD}(G))$			$N_+(\text{FRAC}(G))$			Gruber-Rendl			Dukanovic-Rendl		
n	p	Gap	Gap	CPU-time	Gap	Cuts	CPU-time	Gap	Cuts	CPU-time	Gap	Cuts	CPU-time	Gap	Cuts	CPU-time
150	0.1	32.117	12.954	4.358	12.952	6.4	9.264	11.369	1311.8	17.888	11.356	1352.8	17.240	12.625	40.4	4.692
	0.2	75.793	20.269	2.910	20.269	0.2	3.420	20.109	187.2	5.642	20.102	202.8	5.746	20.259	1.0	1.572
	0.3	82.288	22.330	1.720	22.330	1.0	2.378	22.316	27.0	3.154	22.316	27.8	3.022	22.330	0.0	0.862
	0.4	75.183	28.268	1.144	28.266	6.8	2.632	28.265	5.6	2.456	28.264	6.0	2.476	28.268	0.0	0.880
	0.5	56.020	24.861	1.096	24.362	193.2	3.514	24.860	1.6	2.230	24.860	1.6	2.314	24.861	0.0	1.078
	0.6	39.308	21.161	1.332	17.281	1095.4	10.900	21.161	0.6	1.988	21.161	0.6	1.926	21.161	0.0	0.984
	0.7	23.890	14.027	1.280	6.200	1479.0	15.202	14.026	0.8	2.434	14.026	0.8	2.354	14.027	0.0	1.126
	0.8	14.920	10.140	2.954	3.441	1250.2	14.856	10.137	1.4	4.442	10.137	1.4	4.546	10.140	0.0	2.214
	0.9	8.900	1.378	15.110	1.071	926.0	54.250	1.341	8.0	27.070	1.341	8.0	27.380	1.366	1.0	11.030
175	0.1	37.780	15.091	5.582	15.091	0.6	7.752	14.092	1155.4	20.354	14.080	1182.6	20.218	15.023	6.8	3.458
	0.2	93.892	25.326	3.208	25.326	0.0	3.212	25.274	71.0	6.488	25.272	77.6	6.392	25.326	0.0	1.166
	0.3	87.880	26.152	1.986	26.152	0.0	1.994	26.145	8.8	3.472	26.145	9.4	3.352	26.152	0.0	1.042
	0.4	72.113	27.688	1.384	27.683	6.8	2.890	27.687	1.2	2.260	27.687	1.2	2.308	27.688	0.0	1.076
	0.5	58.318	29.078	1.274	28.505	281.0	4.530	29.077	0.6	1.902	29.077	0.6	1.952	29.078	0.0	1.162
	0.6	32.564	18.100	1.542	13.062	1364.0	18.580	18.100	0.0	1.704	18.100	0.2	2.010	18.100	0.0	1.496
	0.7	23.143	17.776	1.734	6.229	2087.0	19.966	17.775	0.4	2.702	17.775	0.4	2.758	17.776	0.0	1.292
	0.8	7.907	7.531	2.718	1.506	1400.0	14.842	7.530	0.4	4.152	7.530	0.4	3.946	7.531	0.0	2.262
	0.9	9.250	5.341	6.732	4.045	1000.0	18.314	5.338	1.0	10.320	5.338	1.0	10.270	5.341	0.0	4.680
200	0.1	46.957	19.330	6.566	19.330	0.2	8.150	18.723	1018.2	25.144	18.714	1052.4	24.594	19.328	0.2	2.390
	0.2	102.539	27.924	3.756	27.924	0.0	3.762	27.912	29.8	7.956	27.911	31.4	8.010	27.924	0.0	1.414
	0.3	93.334	30.774	2.270	30.774	0.0	2.276	30.773	1.8	3.516	30.773	1.8	3.502	30.774	0.0	1.198
	0.4	73.543	30.550	1.524	30.537	10.0	3.346	30.550	0.4	1.982	30.550	0.4	1.992	30.550	0.0	1.276
	0.5	56.527	30.597	1.516	29.540	634.0	9.378	30.597	0.0	1.704	30.597	0.0	1.738	30.597	0.0	1.362
	0.6	39.889	25.708	1.624	20.138	1587.4	15.810	25.708	0.0	1.880	25.708	0.0	1.918	25.708	0.0	1.546
	0.7	30.543	25.579	2.160	12.494	2294.0	32.826	25.579	0.0	2.434	25.579	0.0	2.466	25.579	0.0	1.594
	0.8	9.720	13.474	3.104	1.836	1400.0	21.840	13.474	0.0	3.426	13.474	0.0	3.472	13.474	0.0	1.918
	0.9	9.600	10.123	5.136	5.665	1800.0	22.542	10.120	0.8	8.992	10.120	0.8	8.854	10.123	0.0	4.036
225	0.1	54.825	22.934	7.556	22.934	0.0	7.564	22.559	798.4	27.196	22.554	836.4	27.424	22.934	0.0	2.506
	0.2	112.700	32.506	4.294	32.506	0.0	4.304	32.503	11.4	9.216	32.503	12.6	9.030	32.506	0.0	1.866
	0.3	97.343	34.482	3.214	34.482	0.2	3.622	34.482	0.6	4.530	34.482	0.6	4.614	34.482	0.0	1.556
	0.4	81.214	38.547	2.134	38.524	20.2	4.282	38.547	0.0	2.370	38.547	0.0	2.412	38.547	0.0	1.682
	0.5	59.427	35.876	1.876	33.947	1088.8	14.466	35.876	0.2	2.504	35.876	0.2	2.546	35.876	0.0	1.702
	0.6	43.044	32.497	1.860	24.985	2000.0	18.024	32.497	0.0	2.212	32.497	0.0	2.256	32.497	0.0	1.744
	0.7	29.957	26.023	1.782	13.745	2825.2	34.120	26.023	0.0	2.232	26.023	0.0	2.288	26.023	0.0	1.992
	0.8	13.733	15.208	2.730	0.149	1914.2	194.072	15.208	0.0	3.172	15.208	0.0	3.176	15.208	0.0	2.194
	0.9	8.130	12.293	5.274	4.074	2600.0	32.476	12.293	0.0	5.812	12.293	0.0	5.794	12.293	0.0	3.908
250	0.1	64.521	27.371	8.426	27.371	0.0	8.442	27.112	615.2	27.514	27.109	641.2	28.054	27.371	0.0	2.978
	0.2	118.348	35.462	5.848	35.462	0.0	5.866	35.459	7.4	11.872	35.459	7.8	11.742	35.462	0.0	2.206
	0.3	103.026	39.136	4.196	39.136	0.0	4.214	39.136	0.4	5.422	39.136	0.4	5.444	39.136	0.0	2.082
	0.4	78.520	38.335	2.440	38.312	24.0	4.770	38.335	0.2	3.216	38.335	0.2	3.274	38.335	0.0	2.076
	0.5	54.333	33.563	1.966	31.421	1239.6	17.622	33.563	0.0	2.412	33.563	0.0	2.446	33.563	0.0	2.298
	0.6	46.178	39.416	2.066	28.532	3000.0	34.146	39.416	0.0	2.646	39.416	0.0	2.664	39.416	0.0	2.212
	0.7	39.114	38.531	1.938	23.678	3000.0	36.856	38.531	0.0	2.594	38.531	0.0	2.612	38.531	0.0	2.476
	0.8	19.867	20.012	2.426	4.433	2500.2	72.364	20.012	0.0	3.104	20.012	0.0	3.100	20.012	0.0	2.870
	0.9	5.950	11.717	5.430	3.232	2400.0	40.546	11.716	0.2	7.422	11.716	0.2	7.380	11.717	0.0	6.352
275	0.1	71.409	29.643	8.868	29.643	0.0	8.896	29.528	305.4	20.656	29.526	318.8	20.270	29.643	0.0	3.324
	0.2	135.226	45.260	5.384	45.260	0.0	5.408	45.258	2.8	11.534	45.258	3.0	11.734	45.260	0.0	2.338
	0.3	107.159	42.996	4.576	42.996	0.0	4.606	42.996	0.4	5.996	42.996	0.4	6.108	42.996	0.0	2.360
	0.4	82.947	43.825	2.596	43.790	42.0	5.052	43.825	0.0	3.102	43.825	0.0	3.176	43.825	0.0	2.320
	0.5	59.050	39.984	2.282	36.995	1609.0	19.978	39.984	0.0	2.892	39.984	0.0	2.956	39.984	0.0	2.604
	0.6	46.384	42.860	2.274	29.635	3000.0	39.284	42.860	0.0	2.976	42.860	0.0	3.026	42.860	0.0	2.640
	0.7	26.504	30.619	2.086	13.342	3600.0	52.436	30.619	0.0	2.926	30.619	0.0	3.020	30.619	0.0	2.780
	0.8	22.100	25.079	3.278	9.048	3000.0	69.560	25.079	0.0	4.068	25.079	0.0	4.094	25.079	0.0	3.322
	0.9	5.920	5.030	15.670	5.000	200.0	18.424	5.030	0.0	16.608	5.030	0.0	16.684	5.030	0.0	45.198
300	0.1	75.661	31.124	10.342	31.124	0.0	10.370	31.047	256.2	23.402	31.046	265.2	24.460	31.124	0.0	3.824
	0.2	137.919	46.238	7.386	46.238	0.0	7.416	46.237	1.4	12.340	46.237	1.4	12.428	46.238	0.0	3.204
	0.3	113.088	48.426	5.288	48.426	0.0	5.330	48.426	0.0	5.694	48.426	0.0	5.862	48.426	0.0	2.654
	0.4	89.040	50.228	3.086	50.166	76.8	6.572	50.228	0.0	3.682	50.228	0.0	3.782	50.228	0.0	2.584
	0.5	62.433	46.061	2.556	41.439	2000.0	23.406	46.061	0.0	3.362	46.061	0.0	3.408	46.061	0.0	2.806
	0.6	42.224	39.956	2.474	26.444	3000.0	42.344	39.956	0.0	3.414	39.956	0.0	3.466	39.956	0.0	2.858
	0.7	23.650	31.619	2.216	11.347	4000.0	67.134	31.619	0.0	3.390	31.619	0.0	3.418	31.619	0.0	3.128
	0.8	23.333	29.809	3.426	11.667	3600.0	69.064	29.809	0.0	4.432	29.809	0.0	4.404	29.809	0.0	3.538
	0.9	7.090	23.160	6.302	5.681	4200.0	97.612	23.160	0.0	7.702	23.160	0.0	7.694	23.160	0.0	4.710

Table 4.1: Numerical results on Erdös–Rényi $G(n, p)$ random graphs

	$\theta^+(G)$		$N_+(\mathrm{NOD}(G))$			$N_+(\mathrm{FRAC}(G))$			Gruber-Rendl			Dukanovic-Rendl		
Graph	Gap	CPU-time	Gap	Cuts	CPU-time	Gap	Cuts	CPU-time	Gap	Cuts	CPU-time	Gap	Cuts	CPU-time
1zc512	9.677	27.65	9.677	1000	60.11	9.677	0	28.01	9.677	0	29.78	9.677	0	17.06
1zc1024	14.286	153.15	14.286	1000	2803.55	-	-	-	-	-	-	-	-	-
brock200_1	29.508	3.53	29.508	0	3.55	29.505	10	6.47	29.505	10	6.79	29.508	0	1.27
brock200_2	17.758	1.59	16.795	659	12.31	17.758	0	1.81	17.758	0	1.86	17.758	0	1.30
brock200_3	24.479	1.70	24.467	7	3.80	24.479	1	3.35	24.479	1	3.43	24.479	0	1.43
brock200_4	24.242	2.64	24.242	1	4.32	24.242	0	2.76	24.242	0	2.79	24.242	0	1.14
brock400_1	45.670	11.79	45.670	0	11.83	45.670	0	12.66	45.670	0	13.00	45.670	0	4.32
brock400_2	35.160	12.16	35.160	0	12.23	35.160	0	12.85	35.160	0	13.29	35.160	0	4.31
brock400_3	26.324	11.83	26.324	0	11.89	26.324	0	12.51	26.324	0	12.89	26.324	0	4.21
brock400_4	18.883	12.29	18.883	0	12.37	18.883	0	12.96	18.883	0	13.36	18.883	0	4.55
brock800_1	82.032	19.27	80.646	2000	64.28	-	-	-	-	-	-	-	-	-
brock800_2	75.435	19.33	73.896	2000	72.00	-	-	-	-	-	-	-	-	-
brock800_3	67.530	19.83	66.043	2000	63.54	-	-	-	-	-	-	-	-	-
brock800_4	61.541	19.17	60.039	2000	63.59	-	-	-	-	-	-	-	-	-
C125-9	10.431	3.72	10.429	14	8.59	8.075	1442	14.38	8.054	1488	14.19	9.597	128	4.40
C250-9	26.856	9.70	26.856	0	9.70	26.618	565	30.45	26.613	610	28.16	26.856	0	2.45
C500-9	46.630	24.08	46.630	0	24.12	46.630	1	53.76	46.630	1	54.76	46.630	0	9.97
DSJC125.1	11.896	3.44	11.881	36	8.26	9.651	1454	13.48	9.624	1570	13.82	10.993	146	7.07
DSJC125.5	14.021	1.16	13.531	141	3.76	14.017	8	2.67	14.016	10	2.94	14.021	0	0.78
DSJC125.9	0.000	1.61	0.000	0	1.67	0.000	0	1.68	0.000	0	1.70	0.000	0	1.10
DSJC500-5	73.621	6.01	58.014	4000	85.91	73.621	0	9.79	73.621	0	10.06	73.621	0	6.66
hamming10-4	6.667	29.41	6.667	0	29.87	-	-	-	-	-	-	-	-	-
johnson32-2-4	0.000	3.45	0.000	0	3.55	-	-	-	-	-	-	-	-	-
keller4	22.417	4.24	22.236	144	12.24	22.388	48	8.15	22.388	48	7.74	22.417	0	1.66
keller5	14.799	81.25	14.799	0	81.49	-	-	-	-	-	-	-	-	-
MANN_a9	9.219	0.61	9.201	504	1.43	6.811	1000	2.96	6.811	1036	9.47	7.087	708	5.22
MANN_a27	5.367	9.26	4.752	1000	76.14	4.057	1212	631.01	4.057	8000	2707.50	5.367	0	5.54
p_hat300-1	25.253	8.98	7.288	3000	120.48	25.253	0	9.95	25.253	0	9.93	25.253	0	15.39
p_hat300-2	6.855	80.75	6.768	727	310.26	6.317	988	206.70	6.314	1056	228.55	6.819	4	141.19
p_hat300-3	13.057	14.65	13.053	75	34.93	12.829	761	28.79	12.826	805	28.99	13.057	0	32.17
p_hat500-1	44.533	17.27	18.721	5000	381.28	-	-	-	-	-	-	-	-	-
p_hat500-2	48.306	190.29	48.225	1000	918.25	47.863	1043	682.15	47.860	1054	715.59	48.300	1	264.72
p_hat500-3	15.622	34.66	15.621	72	82.90	15.524	548	80.94	15.522	574	81.52	15.622	0	65.56
p_hat700-1	36.774	33.90	2.709	7000	2038.55	-	-	-	-	-	-	-	-	-
p_hat700-2	10.091	426.60	10.023	1000	1270.03	-	-	-	-	-	-	-	-	-
p_hat700-3	15.734	77.08	15.733	188	199.38	-	-	-	-	-	-	-	-	-
sanr200_0.7	31.296	2.54	31.296	0	2.57	31.291	3	4.58	31.291	3	4.61	31.296	0	1.62
sanr200_0.9	16.440	6.49	16.440	1	13.67	15.786	1060	27.41	15.774	1083	25.53	16.440	0	2.12
sanr400_0.5	55.217	4.05	46.952	3000	52.79	55.217	0	5.89	55.217	0	6.18	55.217	0	4.34
sanr400_0.7	61.746	7.38	61.746	0	7.46	61.746	0	8.45	61.746	0	8.62	61.746	0	4.34

Table 4.2: Numerical results on DIMACS second challenge's graphs

Chapter 5

Application of $N_+(\cdot)$ operator to $\mathrm{REP}(G)$

The computation of lower bounds, as well as exact methods for the chromatic number $\chi(G)$ of a graph are well researched topics. As we reviewed in Section 3.2, bounds obtained from linear relaxations may be loose, while those coming from semidefinite programming are stronger but they come at a heavier computational cost. Moreover, semidefinite approaches based on the Lovász theta function yield estimates lying below the fractional chromatic number $\chi^f(G)$ and one needs powerful operators and/or tailored algorithms to cross this threshold [19, 33].

As we have seen in Chapter 4, Lovász and Schrijver's Lift-and-Project operator is able to strengthen linear formulations into semidefinite ones, improving the bound's quality of the first. Despite the computational burden yields by this process in general, we were able to mitigate it by the employment of a general purpose algorithm for large-scale SDPs along with a cutting-plane scheme in our application to the MSSP. Although the $N_+(\cdot)$ operator have been thoroughly studied for the stable set problem [10, 16, 53], to the best of our knowledge we are not aware of an application of such operator to the graph coloring.

In this chapter we deal with the following computational study: driven by the interesting results followed by the lifting of the nodal polytope for the stable set problem, we investigate a new SDP relaxation obtained by the application of the Lift-and-Project operator to a compact linear formulation for the graph coloring problem.

As we have seen, the selection of the formulation to which apply the operator $M_+(\cdot)$ is a crucial step. Here, some remarks are needed. For the stable set problem we were able to identify a LP requiring only $|V|$ inequalities (where V is the set of nodes in a graph) and this led to face more compact semidefinite relaxations yield by the lifting. For the GCP the question is more involved: considering the classical assignment formulation `GCP-ASS`, for example, not only the number of constraints, but also the variables become quadratic w.r.t. the size of the graph. Due to this fact, after a preliminary attempt to lift the assignment formulation, we opted for a more compact one proposed by Campelo et al. [12] that we are going to review in the next section.

5.1 Representative formulation

Campelo et al. [12] proposed an alternative linear formulation for the GCP, the so-called *representative formulation*. In order to introduce it, we need to fix some notation. Let $G = (V, E)$ be a simple and connected graph, with $n = |V|$ and $m = |E|$ and let $\bar{G} = (V, \bar{E})$ its complement graph. For $v \in V$, let $\Gamma(v)$ and $\bar{\Gamma}(v)$ denote the neighborhood of v in G and $\bar{G}$ (also referred as anti-neighborhood of v), respectively. For simplicity, we write $\Gamma[v] := \Gamma(v) \cup \{v\}$, the same for $\bar{\Gamma}[v]$. At last, we denote by $E[S]$ the set of edges of $G[S]$, i.e. the subgraph of G induced by the nodes in $S \subseteq V$. The authors pointed out the following remarks:

Remarks 5.1. *A node $u \in V$ is called* universal *if it is connected to all other vertices in V, i.e. $\bar{\Gamma}(u) = \emptyset$. If such a node exists in G, then $\chi(G) = \chi(G[V \setminus \{u\}]) + 1$, since u needs a color for itself only.*

Remarks 5.2. *Given a node $u \in V$, the subset $S \subseteq \bar{\Gamma}(u)$ is a set of* isolated vertices *in $G[\bar{\Gamma}(u)]$ if $E[\bar{\Gamma}(u)] = E[\bar{\Gamma}(u) \setminus S]$, in this case we have that $\chi(G) = \chi(G[V \setminus S])$, since all the nodes in S can be assigned to the same color of u.*

By these remarks they make the following

Assumption 5.3. *G has no universal vertices and the anti-neighborhood of every vertex in G has no isolated vertices.*

Since a vertex coloring divides the vertices into disjoint color classes, Campelo et al. suggested a model in which each color class is represented by exactly one vertex. In order to achieve that, they introduce a binary variable x_{uv} for each non-adjacent pair of vertices $u, v \in V$ stating whether the color of v is represented by u or not, along with additional binary variables x_{uu} indicating if u represents its own class. In particular, the binary variables are defined as follows:

$$\forall\ u \in V,\ v \in \bar{\Gamma}[u],\ \text{let } x_{uv} = \begin{cases} 1 & \text{if } u \text{ represent the color of } v \\ 0 & \text{otherwise} \end{cases}$$

Thus, an exact formulation for the graph coloring on G can be obtained by the following ILP:

$$\chi(G) = \min \sum_{u \in V} x_{uu}$$

$$\sum_{u \in \bar{\Gamma}[v]} x_{uv} \geq 1 \qquad \forall\, v \in V \tag{5.1}$$

$$x_{uv} + x_{uw} \leq x_{uu} \qquad \forall\, u \in V,\ \{v, w\} \in E[\bar{\Gamma}(u)] \tag{5.2}$$

$$x_{uv} \in \{0, 1\} \qquad \forall u \in V, v \in \bar{\Gamma}[u].$$

Constraints (5.1) states that each vertex $u \in V$ in the graph must be represented either by itself or by some vertex not adjacent to u, i.e. for some v in the anti-neighborhood of u. On the other hand, constraints (5.2) enforces adjacent nodes to have different representatives. The authors remarked that by Assumption 5.3, if $u \in V$ and $v \in \bar{\Gamma}(u)$, then the inequality $x_{uv} \leq x_{uu}$ is a consequence of constraints (5.2). Such class of inequalities have the following meaning: if u represents v, then u must be the representative of its color.

The representative formulation described make use of exactly $|V| + 2|\bar{E}|$ binary variables while the number of constraints are $O(|V||\bar{E}|)$, and hence the dimension clearly depends on the density of the graph G. Moreover, we remark that the x_{uv}'s are not symmetric by definition, i.e. $x_{uv} \neq x_{vu}$. Here we focus on the polytope defined by the continuous relaxation of the previous formulation, i.e.

$$\text{REP}(G) = \left\{ x \in [0, 1]^{|V|+2|\bar{E}|} \mid (5.1) \text{ and } (5.2) \text{ hold} \right\}.$$

5.2 Lifting the representative polytope

In this section we describe how to build matrix-cuts from the application of the $M_+(\cdot)$ operator to REP(G), similarly to Section 4.2. We start from recalling the definition of the augmented matrix, let $n = |V| + 2|\bar{E}|$, given a vector $x \in \text{REP}(G)$ in the original space, we index its generic entry by x_{uv} for some $u \in V$ and some $v \in \bar{\Gamma}(u)$. Hence we define

$$Y := \begin{pmatrix} 1 & x^\top \\ x & X \end{pmatrix},$$

with $X \in \mathcal{S}_n^+$, where $(X)_{uv,hk} = (X)_{hk,uv} = x_{uv,hk}$ for some $u, h \in V,\ v \in \bar{\Gamma}[u],\ k \in \bar{\Gamma}[h]$ are the variables in the "quadratic" space with $\text{diag}(X) = x$. Again, to apply Step 1 of the procedure as reported in Section 3.3.2 we need to multiply constraints (5.1) and (5.2) by $x_{hk} \geq 0$ and $1 - x_{hk} \geq 0,\ h \in V, k \in \bar{\Gamma}[h]$. Thus, we are going to report all possible cases of inequalities generated by an exhaustive case analysis. As for Section 4.2, we perform the linearization of the inequalities implicitly (i.e. by the substitutions $x_{uv} = x_{uv}^2$ and $x_{uv,hk} = x_{uv}x_{hk}$ for all uv's and hk's).

Multiplying by $x_{hk} \geq 0$ Starting from (5.1), we need to develop the following product:

$$x_{hk}\left(\sum_{u\in\bar{\Gamma}[v]} x_{uv} - 1\right) \geq 0.$$

Here we can identify only two possible cases, namely:

1. $(h \in \bar{\Gamma}[v],\ k \in V,\ k = v)$, then we obtain

$$\sum_{u\in\bar{\Gamma}[v]\setminus\{h\}} x_{uv,hv} \geq 0 \tag{5.3}$$

2. $(h \notin \bar{\Gamma}[v],\ k \in V,\ k \neq v)$, in this case we obtain

$$\sum_{u\in\bar{\Gamma}[v]} x_{uv,hv} - x_{hk} \geq 0 \tag{5.4}$$

Considering constraints (5.2), the product to develop is

$$x_{hk}\left(x_{uu} - x_{uv} - x_{uw}\right) \geq 0,$$

which is equivalent to

$$x_{uu,hk} - x_{uv,hk} - x_{uw,hk} \geq 0, \tag{5.5}$$

for each $u, h \in V,\ k \in \bar{\Gamma}[h],\ \{v, w\} \in E[\bar{\Gamma}(v)]$. Here we remark that if $hk \in \{uu, uv, uw\}$ the corresponding term in the last equation becomes simply x_{hk}.

Multiplying by $1-x_{hk} \geq 0$ Again, starting from (5.1), we need to develop the following product:

$$(1 - x_{hk})\left(\sum_{u\in\bar{\Gamma}[v]} x_{uv} - 1\right) \geq 0.$$

Here we consider two cases, as before:

1. $(h \in \bar{\Gamma}[v],\ k \in V,\ k = v)$, then we obtain

$$\sum_{u\in\bar{\Gamma}[v]} x_{uv} - \sum_{u\in\bar{\Gamma}[v]\setminus\{h\}} x_{uv,hv} \geq 1 \tag{5.6}$$

2. $(h \notin \bar{\Gamma}[v],\ k \in V)$, in this case we obtain

$$\sum_{u \in \bar{\Gamma}[v]} (x_{uv} - x_{uv,hv}) + x_{hk} \geq 1 \tag{5.7}$$

Considering constraints (5.2), the product to develop is

$$(1 - x_{hk})\,(x_{uu} - x_{uv} - x_{uw}) \geq 0,$$

which is equivalent to

$$x_{uu} - x_{uv} - x_{uw} \geq x_{uu,hk} - x_{uv,hk} - x_{uw,hk} \tag{5.8}$$

for each $u, h \in V$, $k \in \bar{\Gamma}[h]$, $\{v, w\} \in E[\bar{\Gamma}(v)]$. Again, the last inequality may be simplified for some hk. Indeed, when $hk \in \{uu, uv, uw\}$ one of the term on the right-hand side can be elided with the corresponding one to the left-hand side.

As remarked in Lovász and Schrijver [53], since $\mathrm{REP}(G)$ is contained in Q_n, the n-dimensional unit cube, every $Y \in M_+(\mathrm{REP}(G))$ satisfies, in addition

$$x_{uv,hk} \geq 0, \tag{5.9}$$

$$x_{uv,hk} \leq x_{uv}, \tag{5.10}$$

$$x_{uv} + x_{hk} - x_{uv,hk} \leq 1, \quad \text{for all } u, h \in V,\ v \in \bar{\Gamma}[u],\ k \in \bar{\Gamma}[h], \tag{5.11}$$

$$x_{uv} \leq 1, \quad \text{for all } u \in V,\ v \in \bar{\Gamma}[u]. \tag{5.12}$$

As remarked for $M_+(\mathrm{NOD}(G))$, this set of (non-trivial) constraints is the result of the lifting corresponding to the bound constraints $0 \leq x \leq 1$ in $\mathrm{REP}(G)$ (which are assumed to be in the description of the polytope). The bound constraints (5.12) and $x_{uv,hk} \leq 1$ for all uk's and hk's are implied by the condition $Y \succeq 0$.

Hence, the resulting convex set is defined by

$$M_+(\mathrm{REP}(G)) = \left\{ \begin{pmatrix} 1 & x^\top \\ x & X \end{pmatrix} \in \mathcal{S}^+_{n+1} \;\middle|\; \mathrm{diag}(X) = x,\ (5.3)\text{–}(5.11) \text{ hold} \right\},$$

along with the corresponding projection onto the original space

$$N_+(\mathrm{REP}(G)) := \left\{ \mathrm{diag}(X) \;\middle|\; \begin{pmatrix} 1 & x^\top \\ x & X \end{pmatrix} \in M_+(\mathrm{REP}(G)) \right\}.$$

5.3 Numerical results

In this section we report an experimental analysis focused on evaluating the strength of the relaxation $M_+(\mathrm{REP}(G))$. In particular, we compare its optimal value with three reference lower bounds of $\chi(G)$: the classical theta number $\theta(\bar{G})$ computed by using formulation `th-SDP3`, its strengthening $\theta'(\bar{G})$ proposed by Meurdesoif [59] and the fractional chromatic number $\chi^f(G)$. We remark that the latter represents a target value not straightforward to reach with SDP relaxations based on the theta function.

All computations were carried out on an Intel Xeon CPU E5-2698 v4 running at 2.20GHz, equipped with 256GB of RAM, under Linux (Ubuntu 16.04.7).
Instances are solved with the state-of-the-art general purpose solver for large-scale SDPs, namely `SDPNAL+` [82] based on an alternating direction methods of multipliers combined with a semismooth Newton-Conjugate Gradient method. `SDPNAL+` runs with default settings, optimality tolerance set at 10^{-5} and 4 hours time limit. The code employed to formulate the SDPs can be found at `https://github.com/batt95/SDP_MSSP_GCP`.

One major difficulty in experimenting with relaxation $M_+(\mathrm{REP}(G))$ is represented by its size. As a matter of fact, recall that $\mathrm{REP}(G)$ has $n = |V| + 2|\bar{E}|$ variables and $m = O(|V||E|)$ constraints, so according to the Lift-and-Project procedure described in Section 3.3.2 the number of constraints is $O(nm) = O(|E||V|^2 + |\bar{E}||V||E|)$ which is roughly $O(|V|^4)$. Indeed, we expect these formulations to be challenging to address.

Following the promising results obtained on the stable set application, in a preliminary test on $M_+(\mathrm{REP}(G))$ we investigated the employment of the cutting-plane scheme described in Section 4.4. Unfortunately, the results were not satisfactory enough. While previously we were able to start the cutting-plane algorithm from the resolution of the Lovász theta function for the stable set, in this case it is not straightforward anymore since the relation between $M_+(\mathrm{REP}(G))$ and $\theta(\bar{G})$ is not so trivial, also due to the fact that the variable spaces are different. As a matter of facts, for this numerical experience we were starting the cutting-plane scheme with just a small subset of constraints and then subsequently separating most violated constraints at each iteration but the objective value was improving only when almost all constraints were separated by the scheme making the whole algorithm time-consuming.

Although we encourage a further investigation of the separation scheme, due to this ill-behaviour we opted for a different setup throughout the reminder of these experiments. On the positive side, in this preliminary experience we observed that some classes of inequalities defining $M_+(\mathrm{REP}(G))$ were rarely tight at the final solution. Hence, instead of using a separation scheme we decided to feed to `SDPNAL+` solver a relaxation of $M_+(\mathrm{REP}(G))$, precisely, the one defined by inequalities (5.3)–(5.7) and (5.9).

5.3.1 Instances

The numerical experiments are based on the following collections of graphs.

- Graphs from the second DIMACS implementation challenge instances (see [39]), representing standard benchmarks for graph coloring algorithms. We consider small and reasonably dense graphs, which give rise to tractable $M_+(\mathrm{REP}(G))$ relaxations, and show significant gaps $\chi(G) - \theta(\bar{G})$ and $\chi(G) - \chi^f(G)$. This selection resulted in: the triangle free Mycielski graphs `myciel3`, `myciel4` and `myciel5` along with `1-FullIns_3`, `2-Insertions_3`, `3-Insertions_3` and the two larger instances `dsjc125.9` and `r125.1c`.

- The Petersen's graph (shown in Figure 5.1), which belongs to the well-known class of *Kneser graphs*. The interest of this graph to our analysis comes from the fact that $\chi^f(G)$ and the $\chi(G)$ can differ significantly, while $\chi^f(G)$ number is close to the clique number $\omega(G)$ [54].

- Erdös–Rényi random graphs: $G(n, p)$ where $n \in \mathbb{N}$ is the number of nodes in the graph and $p \in [0, 1]$ is the edge probability. In our experience we reported random graphs with 40 vertices used in [59], with $p \in \{0.5, 0.66, 0.75\}$.

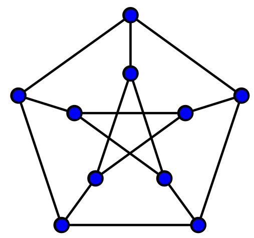


Figure 5.1: The Petersen's graph

5.3.2 Comparison among relaxations

In Table 5.1 optimal values to the SDP relaxations are reported along with $\theta(\bar{G})$, $\theta^+(\bar{G})$ and both the fractional and integer chromatic number (resp. $\chi^f(G)$ and $\chi(G)$). Instances which did not achieve convergence within the time limit of 4 hours are marked with †. For such instances, we anyhow report the objective function values attained at time limit, since their optimality conditions were close to 10^{-4} in most of the cases.

On Table 5.1 a few comments are in order. Consider first the DIMACS instances and the `petersen` graph. Here, $M_+(\mathrm{REP}(G))$ improves over $\theta^+(\bar{G})$ on 7 out of 9 cases and, on the two instances

Graph	$\theta(\bar{G})$	$\theta^+(\bar{G})$	$M_+(\mathrm{REP}(G))$	$\chi^f(G)$	$\chi(G)$
`1-FullIns_3`	3.064	3.064	†3.294	3.333	4
`2-Insertions_3`	2.104	2.104	2.434	2.423	4
`3-Insertions_3`	2.068	2.068	†2.349	2.334	4
`myciel3`	2.400	2.400	3.036	2.900	4
`myciel4`	2.530	2.530	3.267	3.245	5
`myciel5`	2.639	2.639	†3.465	3.553	6
`petersen`	2.500	2.500	2.720	2.500	3
`dsjc125.9`	37.768	37.802	†34.181	42.727	44
`r125.1c`	46.000	46.000	†37.995	46.00	46
$G(40, 0.5)$	6.301	6.325	†6.277	7.030	8
$G(40, 0.66)$	9.260	9.284	†9.126	10.371	11
$G(40, 0.75)$	11.111	11.147	11.146	12.030	13

Table 5.1: Computational experiments on SDP relaxations of the GCP.

`dsjc125.9` and `r125.1c` where $M_+(\mathrm{REP}(G))$ does not outperforms $\theta^+(\bar{G})$, the SDP solver does not converge with the required optimality tolerance. On the 7 instances where the SDP converges with sufficient precision, $M_+(\mathrm{REP}(G))$ attains a very good lower bound, either outperforming $\chi^f(G)$ (on 5 instances out of 7) or nearly reaching $\chi^f(G)$ (on `1-FullIns_3` and `myciel5`). Note also that $M_+(\mathrm{REP}(G))$ closes the integrality gap to zero (by integer rounding) on `myciel3`.

As far as random instances are concerned, $M_+(\mathrm{REP}(G))$ is not able to improve $\theta^+(G)$ but it attains practically the same value whatever the optimality tolerance reached.

It is also important to remark that the evaluation of $M_+(\mathrm{REP}(G))$ by an effective but general purpose SDP solver as `SDPNAL+` requires, on average, a CPU time up to 3 orders of magnitude larger than the one for $\theta(\bar{G})$ and $\theta^+(\bar{G})$. This prevents the straightforward application of $M_+(\mathrm{REP}(G))$ to large graphs.

Chapter 6

On Solving large scale Semidefinite Programs

In the previous chapters we investigated how to obtain strong bounds for two basic CO problems from semidefinite programming by the application of Lovász and Schrijver Lift-and-Project operator. Here we focus more on how to deal with these SDPs in practice. We recall that semidefinite programming can be solved to arbitrary precision in polynomial time [76] using interior-point methods [62], providing a good precision in a reasonable time for small and medium size formulations. Despite this class of algorithms are widely used, as we have seen semidefinite programs of our interest can easily reach several millions of inequality constraints, making interior-point methods impractical both in terms of computation time and memory requirements. As a matter of facts, in our previous computational experiments we used `SDPNAL+` [73], a well-established state-of-the-art `MATLAB` software implementing an Alternating direction methods of multipliers (ADMMs) combined with a semismooth Newton-Conjugate Gradient method. Augmented Lagrangian methods are known to be an alternative to interior point methods and currently represent the most popular first-order algorithms used to handle large scale semidefinite programs [8, 9, 56, 65]. ADMMs, which are a variant of augmented Lagrangian methods, gained growing attention in the last years [17, 73, 77, 82] and their success comes from the fact that as first-order methods avoid computing, storing and factorizing large Hessian matrices and are able to scale to much larger problems than interior point methods. This of course at some cost in accuracy, that should be correctly addressed in case the semidefinite programming problem is the relaxation of a combinatorial problem and one aims at obtaining a valid bound on its optimal solution.

In this chapter we present our computational study on methods for solving semidefinite programs: we devise the ADMM algorithm `ADAL` [56, 65, 77] for solving SDPs in standard form, then extending this method to handle problems with inequality constraints. Hence, we discuss different methodologies to recover a valid dual bound on the optimal primal value, starting from an approximated solution. Then numerical results are reported measuring the performance of our alternative `ADAL` implementation in `python` and `SDPNAL+` on random SDPs as well as MSSP and GCP semidefinite relaxations. In addition, we perform a computational study on `ADAL` equipped with different procedures to compute safe bounds for CO problems, discussing the trade off between quality of the safe bound and computational burden.

6.1 An ADMM method for SDPs with inequality constraints

We focus on SDPs in the following form:

$$\begin{aligned} \min \quad & \langle C, X\rangle \\ \text{s.t.} \quad & \langle A^i, X\rangle \le b_i, \qquad \forall i = 1,\dots,l \\ & \langle A^j, X\rangle = b_j, \qquad \forall j = l+1,\dots,m \\ & X \in \mathcal{S}_n^+ \end{aligned} \tag{6.1}$$

where $C \in \mathcal{S}_n$, $A^i \in \mathcal{S}_n$, $i \in [m+l]$ and $b \in \mathbb{R}^{m+l}$. In order to deal with problem (6.1) a standard way is to add slack variables $s_i \ge 0$, $i \in [l]$ and expand the matrix variable X to $\bar{X} \in \mathcal{S}_{n+l}$:

$$\bar{X} := \begin{pmatrix} X & \mathbf{0}_{n,l} \\ \mathbf{0}_{l,n} & \mathrm{Diag}\,(s) \end{pmatrix}.$$

Recall that if B is a diagonal matrix, the constraint $B \succeq 0$ boils down to $B \ge 0$. In particular, imposing $\bar{X} \succeq 0$ is equivalent to consider $X \succeq 0$ and $s \ge 0$. By expanding the matrices A^i, A^j, and C, $i = 1,\dots,l$; $j = l+1,\dots,m$; to $\bar{A}^i$, $\bar{A}^j$ and $\bar{C}$ as

$$\bar{A}^i := \begin{pmatrix} A^i & \mathbf{0}_{n,l} \\ \mathbf{0}_{l,n} & e_i e_i^\top \end{pmatrix}, \qquad \bar{A}^j := \begin{pmatrix} A^j & \mathbf{0}_{n,l} \\ \mathbf{0}_{l,n} & \mathbf{0}_{l,l} \end{pmatrix}, \qquad \bar{C} := \begin{pmatrix} C & \mathbf{0}_{n,l} \\ \mathbf{0}_{l,n} & \mathbf{0}_{l,l} \end{pmatrix}$$

problem (6.1) can be rewritten as an SDP in standard form as follows:

$$\begin{aligned} \min \quad & \langle \bar{C}, \bar{X}\rangle \\ \text{s.t.} \quad & \bar{\mathcal{A}}(X) = b \\ & \bar{X} \in \mathcal{S}_{n+l}^+ \end{aligned} \tag{6.2}$$

where $b := (b_1, \ldots, b_m) \in \mathbb{R}^m$ and $\bar{\mathcal{A}} : \mathcal{S}_{n+l} \to \mathbb{R}^m$ is the linear operator $(\bar{\mathcal{A}}(X))_i = \langle \bar{A}^i, X \rangle$ with $\bar{A}^i \in \mathcal{S}_{n+l}$, $i \in [m]$. The dual problem of (6.2) is defined as

$$\begin{aligned} \min \quad & b^\top y \\ \text{s.t.} \quad & \bar{\mathcal{A}}^\top(y) + \bar{Z} = \bar{C} \\ & \bar{Z} \in \mathcal{S}^+_{n+l}, \end{aligned} \tag{6.3}$$

where $\bar{\mathcal{A}}^\top : \mathbb{R}^m \to \mathcal{S}_{n+l}$ is the adjoint operator of $\bar{\mathcal{A}}$, namely $\bar{\mathcal{A}}^\top y = \sum_i y_i \bar{A}^i$ for $y \in \mathbb{R}^m$. Note that the matrix $\bar{Z} \in \mathcal{S}_{n+l}$ is a "surplus" matrix variable that can be written as

$$\bar{Z} := \begin{pmatrix} Z & \mathbf{0}_{n,l} \\ \mathbf{0}_{n,l} & \operatorname{diag}(p) \end{pmatrix},$$

with $p \in \mathbb{R}^l$. In particular, the equality constraint in (6.3) can be rewritten as

$$\bar{C} - \bar{\mathcal{A}}^\top(y) - \bar{Z} = \begin{pmatrix} C - \mathcal{A}^\top(y) - Z & & \mathbf{0}_{n,l} & \\ & -y_1 - p_1 & & \\ \mathbf{0}_{n,l} & & \ddots & \\ & & & -y_l - p_l \end{pmatrix} = 0.$$

Assuming that both the primal (6.2) and the dual (6.3) problems have strictly feasible points (i.e. Slater's condition is satisfied) strong duality holds and $(y, \bar{Z}, \bar{X})$ is optimal for (6.2) and (6.3) if and only if the following conditions hold:

$$\bar{\mathcal{A}}(\bar{X}) = b, \qquad \bar{\mathcal{A}}^\top(y) + \bar{Z} = \bar{C}, \qquad \bar{Z}\bar{X} = 0, \qquad \bar{X} \in \mathcal{S}^+_{n+l}, \qquad \bar{Z} \in \mathcal{S}^+_{n+l}. \tag{6.4}$$

In the following, we assume that the constraints formed through the operator $\bar{\mathcal{A}}$ are linearly independent.

6.1.1 `ADAL`: an ADMM for SDPs in standard form

We now present `ADAL` [65, 77], an alternating direction method of multipliers (ADMM) to address standard SDPs and in particular, we extend it in order to be able to deal with problem (6.2). Recently Zhao and Wiegele [79] independently presented a related work in which an extended ADMM in order to solve semidefinite programs arising from graph partitioning problems is proposed. The method we consider is based on the maximization of the augmented Lagrangian built over the dual problem. Let $\bar{X} \in \mathcal{S}_{n+l}$ be the Lagrange multiplier for the dual equation $\bar{\mathcal{A}}^\top(y) + \bar{Z} - \bar{C} = 0$

and $\sigma > 0$ be fixed. The augmented Lagrangian of the dual (6.3) is defined as

$$L_\sigma(y, \bar{Z}; \bar{X}) = b^T y - \langle \bar{\mathcal{A}}^\top(y) + \bar{Z} - \bar{C}, \bar{X} \rangle - \frac{\sigma}{2}\|\bar{\mathcal{A}}^\top(y) + \bar{Z} - \bar{C}\|^2.$$

In augmented Lagrangian methods applied to the dual (6.3) the problem

$$\begin{aligned} \max \quad & L_\sigma(y, \bar{Z}; \bar{X}) \\ \text{s.t.} \quad & y \in \mathbb{R}^m, \quad \bar{Z} \in \mathcal{S}^+_{n+l}, \end{aligned} \tag{6.5}$$

where $\bar{X}$ is fixed and $\sigma > 0$ is a penalty parameter, is addressed at every iteration. When the maximization of the augmented Lagrangian $L_\sigma(y, \bar{Z}; \bar{X})$ is performed by optimizing first with respect to y and then with respect to $\bar{Z}$, we are considering the well known alternating direction method of multipliers (ADMM), first proposed in [56, 65] and then extended in [77]. In the following, we refer to this method as `ADAL`. To be more precise, the new point $(y^{k+1}, \bar{Z}^{k+1}, \bar{X}^{k+1})$ is computed by the following steps:

$$y^{k+1} = \operatorname*{argmax}_{y \in \mathbb{R}^m} L_{\sigma^k}(y, \bar{Z}^k; \bar{X}^k), \tag{6.6}$$

$$\bar{Z}^{k+1} = \operatorname*{argmax}_{Z \in \mathcal{S}^+_n} L_{\sigma^k}(y^{k+1}, \bar{Z}; \bar{X}^k), \tag{6.7}$$

$$\bar{X}^{k+1} = \bar{X}^k + \sigma^k(\mathcal{A}^\top(y^{k+1}) + \bar{Z}^{k+1} - \bar{C}). \tag{6.8}$$

The update of y in (6.6) can be performed in closed form, as it derives from the first-order optimality conditions of the problem on the right-hand side of (6.6): y^{k+1} is the unique solution of

$$\nabla_y L_{\sigma^k}(y, \bar{Z}^k; \bar{X}^k) = b - \bar{\mathcal{A}}(\bar{X}^k + \sigma^k(\bar{\mathcal{A}}^\top y + \bar{Z}^k - \bar{C})) = 0,$$

that is

$$y^{k+1} = (\bar{\mathcal{A}}\bar{\mathcal{A}}^\top)^{-1}\Big(\frac{1}{\sigma^k} b - \bar{\mathcal{A}}(\frac{1}{\sigma^k}\bar{X}^k + \bar{Z}^k - \bar{C})\Big).$$

The update of $\bar{Z}$ in (6.7) is conducted by considering the equivalent problem

$$\min_{\bar{Z} \in \mathcal{S}^+_{n+l}} \|\bar{Z} + W^{k+1}\|^2, \tag{6.9}$$

where

$$W^{k+1} = \frac{\bar{X}^k}{\sigma^k} - \bar{C} + \bar{\mathcal{A}}^\top(y^{k+1}).$$

Solving problem (6.9), is equivalent to project $W^{k+1} \in \mathcal{S}_{n+l}$ onto the (closed convex) cone $\mathcal{S}^-_{n+l}$ and take its additive inverse (see Algorithm 2). Such a projection is computed via the spectral decomposition of the matrix W^{k+1}. Finally, it is easy to see that the update of $\bar{X}$ in (6.8) can be performed considering the projection of $W^{k+1} \in \mathcal{S}_{n+l}$ onto $\mathcal{S}^+_{n+l}$ multiplied by σ^k, namely

$$\begin{aligned}
\bar{X}^{k+1} &= \bar{X}^k + \sigma^k(\bar{\mathcal{A}}^\top(y^{k+1}) + \bar{Z}^{k+1} - \bar{C}) = \\
&= \sigma^k(\bar{X}^k/\sigma^k - \bar{C} + \bar{\mathcal{A}}^\top(y^{k+1}) - (\bar{X}^k/\sigma^k - \bar{C} + \bar{\mathcal{A}}^\top(y^{k+1}))_-) = \\
&= \sigma^k(\bar{X}^k/\sigma^k - \bar{C} + \bar{\mathcal{A}}^\top(y^{k+1}))_+.
\end{aligned}$$

We report in Algorithm 2 the scheme of `ADAL`. `ADAL` is stopped as soon as the following errors related

Algorithm 2 Scheme of `ADAL` from [77]

1: Choose $\sigma > 0$, $\varepsilon > 0$, $\bar{X} \in \mathcal{S}^+_{n+l}$, $\bar{Z} \in \mathcal{S}^+_{n+l}$
2: $\delta = \max\{r_P, r_D\}$
3: **while** $\delta > \varepsilon$ **do**
4: $\quad y = (\bar{\mathcal{A}}\bar{\mathcal{A}}^\top)^{-1}\left(\frac{1}{\sigma}b - \bar{\mathcal{A}}(\frac{1}{\sigma}\bar{X} - \bar{C} + \bar{Z})\right)$
5: $\quad \bar{Z} = -(\bar{X}/\sigma - \bar{C} + \bar{\mathcal{A}}^\top(y))_-$ and $\bar{X} = \sigma(\bar{X}/\sigma - \bar{C} + \bar{\mathcal{A}}^\top(y))_+$
6: $\quad \delta = \max\{r_P, r_D\}$
7: $\quad$ Update σ
8: **end while**

to primal feasibility ($\bar{\mathcal{A}}\bar{X} = b$) and dual feasibility ($\bar{\mathcal{A}}^\top(y) + \bar{Z} + \bar{S} = \bar{C}$) are below a certain accuracy

$$r_P = \frac{\|\bar{\mathcal{A}}\bar{X} - b\|}{1 + \|b\|}, \qquad r_D = \frac{\|\bar{\mathcal{A}}^\top(y) + \bar{Z} - \bar{C}\|}{1 + \|\bar{C}\|}.$$

More precisely, the algorithm stops as soon as the quantity $\delta = \max\{r_P, r_D\}$ is less than a fixed precision $\varepsilon > 0$. The other optimality conditions (namely $\bar{X} \in \mathcal{S}^+_{n+l}$, $\bar{Z} \in \mathcal{S}^+_{n+l}$, $\bar{Z}\bar{X} = 0$) are satisfied up to machine accuracy throughout the algorithm thanks to the projections employed in `ADAL`. The numerical performance of ADMMs, including the one of `ADAL`, strongly depends on the update rule used for the penalty parameter σ. As in [14, 78], we follow the strategy by Lorenz and Tran-Dinh [51], considering at every iteration k the ratio between the norm of the primal variable $\bar{X}^k$ and norm of the dual variable $\bar{Z}^k$.

The memory required to store the augmented matrices $\bar{C}, \bar{A}_i, \bar{A}_j, \bar{Z}$ and $\bar{X}$ gets large with the number l of inequalities and even using efficient sparse matrix implementations may be insufficient to computationally deal with large scale problems. Our idea is then to rewrite the steps of `ADAL` in terms of the original matrices C, A^i and X, so that one can keep in memory only the matrices that are actually defining the problem. Indeed, let $1 \leq i \leq l$ be a generic index of an inequality constraint and let $l + 1 \leq j \leq m$ be a generic index of an equality constraint, then the following holds:

$$
\begin{aligned}
\langle \bar{A}^i, \bar{X} \rangle &= \langle A^i, X \rangle + s_i, \\
\langle \bar{A}^j, \bar{X} \rangle &= \langle A^j, X \rangle, \\
\langle \bar{C}, \bar{X} \rangle &= \langle C, X \rangle;
\end{aligned}
$$

The linear map applied to $\bar{X}$ becomes:

$$
\bar{\mathcal{A}}(\bar{X}) = \begin{pmatrix} \langle A^1, X \rangle \\ \vdots \\ \langle A^l, X \rangle \\ \langle A^{l+1}, X \rangle \\ \vdots \\ \langle A^m, X \rangle \end{pmatrix} + \begin{pmatrix} s^\top \\ \mathbf{0}_{m-l} \end{pmatrix} = \mathcal{A}(X) + \begin{pmatrix} s^\top \\ \mathbf{0}_{m-l}. \end{pmatrix}
$$

Similarly, the adjoint operator $\bar{\mathcal{A}}^T : \mathbb{R}^m \to \mathcal{S}_{n+l}$ of $\bar{\mathcal{A}}$ is defined as

$$
\bar{\mathcal{A}}^T(y) := \sum_m^{i=1} y_i \bar{A}^i = \begin{pmatrix} \sum_m^{i=1} y_i A^i & & \mathbf{0}_{n,l} & \\ & y_1 & & \\ \mathbf{0}_{n,l} & & \ddots & \\ & & & y_l \end{pmatrix} = \begin{pmatrix} \mathcal{A}^\top(y) & & \mathbf{0}_{n,l} & \\ & y_1 & & \\ \mathbf{0}_{n,l} & & \ddots & \\ & & & y_l \end{pmatrix}.
$$

Using the operator $\mathrm{vec}(\cdot)$, we can write $\bar{\mathcal{A}}(\bar{X}) = b$ as $\bar{A}\ \mathrm{vec}(\bar{X}) = b$, where

$$
\bar{A} := \left(\mathrm{vec}(\bar{A}^1), \ldots, \mathrm{vec}(\bar{A}^m)\right)^T \in \mathbb{R}^{m \times (n+l)^2}.
$$

Note that matrix $\bar{A}^i$, $i = 1, \ldots, l$, corresponding to the i-th inequality constraint, is the unique matrix having 1 in position $(n+i, n+i)$. Then, $\bar{A}\bar{A}^\top$ can be expressed in terms of $AA^\top$ as follows:

$$
\bar{A}\bar{A}^\top = AA^\top + \mathrm{diag}\begin{pmatrix} \mathbf{1}_l \\ \mathbf{0}_{m-l} \end{pmatrix},
$$

as the zero entries of $\bar{A}^i$ $i = 1, \ldots, m$, do not contribute in the row-by-column product and the 1 in position $(n+i, n+i)$ contributes only to the entry where $\mathrm{vec}(\bar{A}^i)$ is multiplied by itself, i.e., in position (i,i) of $\bar{A}\bar{A}^\top$. According to the notation introduced, the update of the y variable can be rewritten as follows:

$$y^{k+1} = \left(AA^\top + \text{Diag}\left(\begin{pmatrix} \mathbf{1}_l \\ \mathbf{0}_{m-l} \end{pmatrix} \right) \right)^{-1} \left(\frac{1}{\sigma^k} b - A \text{ vec}\left(\frac{1}{\sigma^k} X^k - C + Z^k \right) + \begin{pmatrix} \frac{1}{\sigma^k} {s^k}^\top + {p^k}^\top \\ \mathbf{0}_{m-l} \end{pmatrix} \right)$$

Furthermore, the spectral decomposition of the matrix W, needed for updating the variable $\bar{X}$ and $\bar{Z}$ can be computed in a "block-wise" fashion. At a generic iteration of `ADAL`, matrix W can be written as follows:

$$W^{k+1} = \begin{pmatrix} \frac{X^k}{\sigma^k} - C + \mathcal{A}^\top(y^{k+1}) & \mathbf{0}_{n,l} \\ \mathbf{0}_{n,l} & \text{Diag}\left(\frac{{s^k}^\top}{\sigma^k} + \begin{pmatrix} y_1^{k+1} \\ \vdots \\ y_l^{k+1} \end{pmatrix} \right) \end{pmatrix}.$$

Then, in order to compute the eigenvalues and eigenvectors of W^{k+1}, we can first compute the spectral decomposition of the matrix $\frac{X^k}{\sigma^k} - C + \mathcal{A}^\top y^{k+1}$, then we trivially get the eigenvalues and eigenvectors of the diagonal part of W^{k+1} and eventually we adjust the dimension of the eigenvectors computed, in order to have them in $\mathbb{R}^{n+l}$.

The convergence of the scheme introduced is inherited by the convergence of `ADAL` [77]. In particular, Algorithm 2 can be interpreted as a fixed point method and we can state the following result

Theorem 6.1. *The sequence* $\{(\bar{X}^k, y^k, \bar{Z}^k)\}$ *generated by Algorithm 2 from any starting point* $(\bar{X}^0, y^0, \bar{Z}^0)$ *converges to a solution* $(\bar{X}^*, y^*, \bar{Z}^*) \in \Omega^*$*, where* Ω^* *is the set of primal and dual solutions of* (6.2) *and* (6.3)*.*

ADMM framework turned out to be quite general. As a matter of fact, many extensions of the basic scheme shown in Algorithm 2 have been presented in literature (see e.g. [14, 77, 79]). In particular, Cerulli et al. [14] deployed a factorization of the dual variable to improve the convergence of the ADMM for solving semidefinite problems with nonnegativity constraints. In the context of ADMMs defined over the dual problem, bounds on the matrix variable can be handled by introducing a further step, where a projection onto the nonnegative orthant is performed. Although these 3-blocks ADMMs may not theoretically converge [15], they perform well in practice.

6.2 Computing valid bounds for semidefinite programs

Despite being suited for solving semidefinite programs with a large number of constraints, ADMMs as first-order methods may require more time to compute high accuracy optimal solutions, compared to interior-point methods. Moreover, conversely to LP methods such as the simplex algorithm for example, where at each iteration a (primal or dual) feasible solution is computed, in ADMMs primal and dual feasibility are satisfied to the desired precision (or close to it) only at the last iterations of the algorithm and then we are unable to retrieve feasible solutions in general, even of moderate quality.

SDPs are used to bound combinatorial problems, hence methodologies able to compute valid bounds - even of moderate quality - have a twofold purpose: from the one hand, as post-processing procedure, they can correct the inaccuracy of the optimal value obtained from the ADMM, from the other hand one can use these procedures through the iterations of the SDP solver when considering, for example, branch-and-bound frameworks to define exact solution methods for specific combinatorial optimization problems.

6.2.1 Valid dual bounds

Given a pair of primal-dual SDPs, weak and strong duality hold under the assumption that both problems are strictly feasible. Duality results imply that the objective function value of every feasible solution of the dual SDP is a valid bound on the optimal objective function value of the primal. Therefore, every dual feasible solution and in particular the optimal dual solution of an SDP relaxation, yields a valid bound on the solution of the related combinatorial optimization problem. Following ideas developed in Cerulli et al. [14], we define a post-processing procedure for `ADAL` on SDPs in form (6.1), that allows to get a feasible dual solution starting from a positive semidefinite matrix $\tilde{Z} \in \mathcal{S}_n^+$. Let $\mathcal{A}_{ineq}$ and $\mathcal{A}_{eq}$ be the linear operators defining the inequality and equality constraints in problem (6.1): $\mathcal{A}_{ineq} = \langle A^i, X \rangle$ with $A^i \in \mathcal{S}_n,\ i = 1, \dots, l$ and $\mathcal{A}_{eq} = \langle A^j, X \rangle$ with $A^j \in \mathcal{S}_n,\ j = l+1, \dots, m$. Let b_{ineq} and b_{eq} be the right hand side vectors accordingly defined. Introducing the adjoint operators of $\mathcal{A}_{ineq}$ and $\mathcal{A}_{eq}$, the dual problem (6.3) can be equivalently written as

$$
\begin{aligned}
\max \quad & -b_{ineq}^\top \lambda + b_{eq}^\top \mu \\
\text{s.t.} \quad & C + \mathcal{A}_{ineq}^\top(\lambda) - \mathcal{A}_{eq}^\top(\mu) = Z \\
& Z \in \mathcal{S}_n^+,\ \lambda \geq 0,
\end{aligned}
\tag{6.10}
$$

with $\lambda \in \mathbb{R}^l$ and $\mu \in \mathbb{R}^{m-l}$. We can then extend the results proposed in [14] and define a procedure to get feasible solutions of problem (6.10) and then, by weak duality, valid bounds on the optimal objective function value of the primal (6.1). Recently, a similar approach have been investigated in

Zhao and Wiegele [79]. Let $\tilde{Z} \in \mathcal{S}_n^+$. If the linear programming problem

$$\begin{aligned} \max \quad & -b_{ineq}^T\lambda + b_{eq}^T\mu \\ \text{s.t.} \quad & C + \mathcal{A}_{ineq}^\top(\lambda) - \mathcal{A}_{eq}^\top(\mu) = \tilde{Z} \\ & \lambda \geq 0 \end{aligned} \tag{6.11}$$

has an optimal solution $(\tilde{\lambda}, \tilde{\mu}) \in \mathbb{R}^m$, then $(\tilde{\lambda}, \tilde{\mu}, \tilde{Z})$ is a feasible solution for (6.10) and the value $-b_{ineq}^\top\tilde{\lambda} + b_{eq}^\top\tilde{\mu}$ is giving a dual bound. If (6.11) is unbounded, then also (6.10) is unbounded and hence the primal (6.1) is not feasible. If (6.11) is infeasible, it means that the starting $\tilde{Z} \in \mathcal{S}_n^+$ does not allow to find a feasible dual solution and then get a dual bound. From a practical point of view, once problem (6.1) is approximately solved by `ADAL`, we can try to get a feasible solution of problem (6.10), by addressing problem (6.11). This is what we have implemented, using GUROBI 9.1.1 [31] as solver for problem (6.11).

6.2.2 Rigorous error bounds

An alternative method to produce safe underestimates for some SDP relaxation has been proposed by Jansson et al. [38] and further adapted within ADMMs framework in [14, 79]. The main idea is based on Lemma 3.1 and Theorem 3.2 in [38] which in our setup, according to the primal-dual pair (6.2)-(6.3), can be stated as follows:

Lemma 6.2. *Let $\bar{Z}, \bar{X} \in \mathcal{S}_{n+l}$ symmetric matrices such that $0 \leq \lambda(\bar{X}) \leq \tilde{x}$. Then*

$$\langle \bar{X}, \bar{Z} \rangle \geq \sum_{i:\lambda_i(\bar{Z})<0} \tilde{x}\lambda_i(\bar{Z}).$$

Theorem 6.3. *Let $\bar{X}^*$ be an optimal solution to (6.2) and let p^* be its optimal value. Given $y \in \mathbb{R}^m$, set*

$$\tilde{Z} = \bar{C} - \bar{\mathcal{A}}^\top(y).$$

Assume we know $\tilde{x} \in \mathbb{R}$ such that $\lambda_{\max}(\bar{X}^*) \leq \tilde{x}$, then the following inequality holds:

$$p^* \geq b^\top y + \sum_{i:\lambda_i(\tilde{Z})<0} \tilde{x}\lambda_i(\tilde{Z}).$$

Proof. Let $\bar{X}^*$ be an optimal solution to (6.2). Then

$$\langle \bar{C}, \bar{X}^* \rangle - b^\top y = \langle \bar{C}, \bar{X}^* \rangle - \langle \bar{\mathcal{A}}(\bar{X}^*), y \rangle = \left\langle C - \bar{\mathcal{A}}^\top(y), \bar{X}^* \right\rangle = \left\langle \tilde{Z}, \bar{X}^* \right\rangle.$$

Hence, Lemma 6.2 implies

$$p^* = \langle \bar{C}, \bar{X}^* \rangle = b^\top y + \left\langle \tilde{Z}, \bar{X}^* \right\rangle \geq b^\top y + \sum_{i:\lambda_i(\tilde{Z})<0} \tilde{x}\lambda_i(\tilde{Z}).$$

□

In order to apply Theorem 6.3 we need to provide an upper bound $\tilde{x}$ on the maximum eigenvalue of the optimal solution $\bar{X}^*$. For structured SDPs, such as semidefinite relaxation of combinatorial problems, such a value might be known. Furthermore, it is well-known that

$$\mathrm{tr}(A) = \sum_i \lambda_i(A), \tag{6.12}$$

for any matrix A. Hence, any SDP bounding from above (or fixing to some value) the trace of the matrix variable X gives us for free the upper bound $\tilde{x}$. This is the case for the Lovász theta function formulation `th-SDP2`, where $\tilde{x} = 1$. Hence, when solving this kind of SDPs the error bound procedure may be applied not only as a post-processing, but also during the iterations of the ADMM and stop the latter prematurely as soon as the quality of the safe bound is satisfactory. At last, we remark that the predominant computational burden yield by the Theorem 6.3 resides in finding the eigenvalues of $\tilde{Z}$.

6.2.3 Norm dual bounds

What we are going to present in this section is a joint work with Jan Schwiddessen from Alpen-Adria-Universität in Klagenfurt and is inspired by the idea of the forthcoming paper [70]. As we have seen, many 0-1 linear programs admits also a semidefinite exact formulation. Usually such exact SDPs, as for the MSSP for example, involve a rank-1 constraint on the matrix variable X. We consider the optimization problem in the form

$$\begin{aligned} \max \quad & \langle C, X \rangle \\ \text{s.t.} \quad & \mathcal{A}(X) = b, \\ & X \succeq 0, \\ & \mathrm{rank}(X) = 1. \end{aligned} \tag{P}$$

A semidefinite relaxation for (P) is obtained by dropping the rank-one constraint:

$$\begin{aligned} \max \quad & \langle C, X \rangle \\ \text{s.t.} \quad & \mathcal{A}(X) = b, \\ & X \succeq 0, \end{aligned} \tag{PSDP}$$

The dual problem of (PSDP) is

$$\begin{aligned} \min \quad & b^\top \mu \\ \text{s.t.} \quad & C - \mathcal{A}^\top(\mu) + Z = 0, \\ & \mu \text{ free}, Z \succeq 0, \end{aligned} \tag{DSDP}$$

Many solution approaches, especially first-order methods, compute dual solutions (μ, Z) that are close to dual feasibility, but that are actually not feasible for (DSDP). Hence again, some post-processing routines are required to obtain valid dual/upper bounds for (P). In the following we describe an idea how we can use any pair of dual variables (μ, Z) with $Z \succeq 0$ to derive valid dual bounds. Suppose that we know an upper bound U on the norm of an optimal solution matrix X of (P):

$$\|X\|_F \leq U.$$

Thus, (P) is equivalent to

$$\begin{aligned} \max \quad & \langle C, X \rangle \\ \text{s.t.} \quad & \mathcal{A}(X) = b, \\ & X \succeq 0, \\ & \operatorname{rank}(X) = 1, \\ & \|X\|_F^2 \leq U^2. \end{aligned} \tag{P-norm}$$

Again, a relaxation of (P) and (P-norm) is obtained after dropping the rank-1 constraint:

$$\begin{aligned} \max \quad & \langle C, X \rangle \\ \text{s.t.} \quad & \mathcal{A}(X) = b, \\ & X \succeq 0, \\ & \|X\|_F^2 \leq U^2. \end{aligned} \tag{SDP-norm}$$

A more convenient way to write the problem is

$$\begin{aligned}
\max \quad & \langle C, X\rangle \\
\text{s.t.} \quad & \mathcal{A}(X) - b && = 0 \\
& -X && \preceq 0 \\
& \|X\|_F^2 - U^2 && \leq 0
\end{aligned}$$

Dualizing all constraints yields the Lagrangian

$$\begin{aligned}
L(X;\mu,Z,\alpha) &=\coloneqq \langle C, X\rangle - \mu^\top(\mathcal{A}(X) - b) - \langle Z, -X\rangle - \frac{\alpha}{2}(\|X\|_F^2 - U^2) \\
&= b^\top\mu + \frac{\alpha}{2}U^2 - \frac{\alpha}{2}\|X\|_F^2 + \left\langle C - \mathcal{A}^\top(\mu) + Z, X\right\rangle.
\end{aligned}$$

The dual function is

$$f(\mu,Z,\alpha) \coloneqq \sup_{X\in\mathcal{S}^n} L(X;\mu,Z,\alpha)$$

and the dual problem is

$$\inf_{\mu \text{ free},\, Z\succeq 0,\, \alpha\geq 0} f(\mu,Z,\alpha).$$

We fix $\alpha > 0$ and consider the dual function

$$f^\alpha(\mu,Z) \coloneqq \max_{X\in\mathcal{S}_n} L(X;\mu,Z,\alpha)$$

The Lagrangian as a function of X is strongly concave. Therefore, the maximum over $\mathcal{S}_n$ is attained and the unique optimizer is characterized by

$$-\alpha X + C - \mathcal{A}^\top(\mu) + Z = 0.$$

Hence we have

$$X^* = \frac{1}{\alpha}\left(C - \mathcal{A}^\top(\mu) + Z\right).$$

We obtain

$$\begin{aligned}
f^\alpha(\mu,Z) &= b^\top\mu + \frac{\alpha}{2}U^2 - \frac{\alpha}{2}\left\|\frac{1}{\alpha}\left(C - \mathcal{A}^\top(\mu) + Z\right)\right\|^2 \\
&\quad + \left\langle C - \mathcal{A}^\top(\mu) + Z, \frac{1}{\alpha}\left(C - \mathcal{A}^\top(\mu) + Z\right)\right\rangle \\
&= b^\top\mu + \frac{\alpha}{2}U^2 + \left(-\frac{\alpha}{2}\frac{1}{\alpha^2} + \frac{1}{\alpha}\right)\left\|C - \mathcal{A}^\top(\mu) + Z\right\|^2 \\
&= b^\top\mu + \frac{\alpha}{2}U^2 + \frac{1}{2\alpha}\left\|C - \mathcal{A}^\top(\mu) + Z\right\|^2.
\end{aligned}$$

We have

$$\mathrm{opt}(P) \leq b^\top \mu + \frac{\alpha}{2} U^2 + \frac{1}{2\alpha} \left\| C - \mathcal{A}^\top(\mu) + Z \right\|^2 \tag{$*$}$$

for any μ, $Z \succeq 0$, $\alpha > 0$ by weak duality.

Now, let's assume that we are given any μ, $Z \succeq 0$. What is the best bound that we can get with these dual variables, i.e., what value of $\alpha > 0$ yields the best bound?

The optimal α is the solution of

$$\begin{array}{rl} \min & \frac{\alpha}{2} U^2 + \frac{1}{2\alpha} K \\ \text{s.t.} & \alpha > 0, \end{array}$$

where $K = \left\| C - \mathcal{A}^\top(\mu) + Z \right\|^2$. The unique minimizer is characterized by

$$\frac{1}{2} U^2 - \frac{K}{2} \frac{1}{\alpha^2} \overset{!}{=} 0.$$

We obtain

$$\alpha^* = \frac{\sqrt{K}}{U} = \frac{1}{U} \left\| C - \mathcal{A}^\top(\mu) + Z \right\|.$$

Thus, we can substitute α^* into $(*)$ and obtain

$$\mathrm{opt}(P) \leq b^\top \mu + U \left\| C - \mathcal{A}^\top(\mu) + Z \right\|. \tag{6.13}$$

Now let's discuss how we can apply what we introduced in practice. Clearly we need to identify a valid upper bound U on the norm of an *optimal* solution of (PSDP) which for well-structured SDPs this may be the case, as we are going to discuss in the next section. We remark that to compute (6.13), the most expensive step is the computation of a norm which value is already calculated by the ADMM as one of the stopping criteria. Hence, this means that the computational effort is negligible and (6.13) may be computed at every single iteration, in principle.

6.3 Numerical results

In this section we report our computational study: we compare the performance of `ADAL` and `SDPNAL+` [82] on randomly generated instances and on instances from SDP relaxations of the stable set problem and the graph coloring problem. Moreover, for these combinatorial optimization problems we are going to discuss also the employment of three methodologies to compute safe bounds, namely Dual bound, Error bound and Norm bound (Sections 6.2.1, 6.2.2 and 6.2.3 respectively). `SDPNAL+` implements an ADMM combined with a semismooth Newton-Conjugate Gradient method. `SDPNAL+` is implemented in MATLAB, with some subroutines in C language incorporated via Mex files. For our comparison, we considered the version of `SDPNAL+` available at `https://blog.nus.edu.sg/mattohkc/softwares/sdpnalplus/`. Our version of `ADAL` is implemented in `python` and uses `NumPy` library, the fundamental package for scientific computing (see `https://numpy.org/`). Our code and instances as well, can be found at `https://github.com/batt95/ADAL-ineq`. In the implementation of `SDPNAL+` a refined management of the matrices is implemented exploiting their symmetry and allows the optimization of the subroutines used throughout the application of the algorithm.

The experiments were carried out on an `Intel Xeon CPU E5-2698 v4` running at `2.20GHz`with `256GB` of `RAM`, under `Linux` (`Ubuntu 16.04.7`).

We compare the performance of the algorithms using performance profiles as proposed by Dolan and Moré [18]. Given a set of solvers $\mathcal{S}$ and a set of problems $\mathcal{P}$, the performance of a solver $s \in \mathcal{S}$ on problem $p \in \mathcal{P}$ is compared against the best performance obtained by any solver in $\mathcal{S}$ on the same problem. The performance ratio is defined as

$$r_{p,s} = t_{p,s} / \min\{t_{p,s'} \mid s' \in \mathcal{S}\},$$

where $t_{p,s}$ is the measure we want to compare, and we consider a cumulative distribution function

$$\rho_s(\tau) = |\{p \in \mathcal{P} \mid r_{p,s} \le \tau\}| / |\mathcal{P}|.$$

The performance profile for $s \in S$ is the plot of the function ρ_s. Being aware of the issue to assess the relative performance of the solvers noted in Gould and Scott [26], we will mainly focus on $\rho_s(1)$ for some solver s, to measure the number of instances where s outperforms other solvers.

6.3.1 Comparison on randomly generated SDPs

The random instances considered in the first experiment are obtained from the instance generator used in [56]. Given n, m and a percentage p, we built 5 instances having $\lfloor pm \rfloor$ number of inequalities. In Table 6.1, we report the comparison between `ADAL` and `SDPNAL+` in terms of number of iterations and CPU time needed in order to reach an accuracy of 10^{-5}. We consider instances with $n \in \{200, 250, 500, 1000\}$, $m \in \{5000, 10000, 25000, 50000, 100000\}$ and $p \in \{0.25, 0.5, 0.75\}$. We excluded those combinations of n and m leading to matrices A_i with linearly dependent rows. We set a time limit of 1800 seconds CPU time.

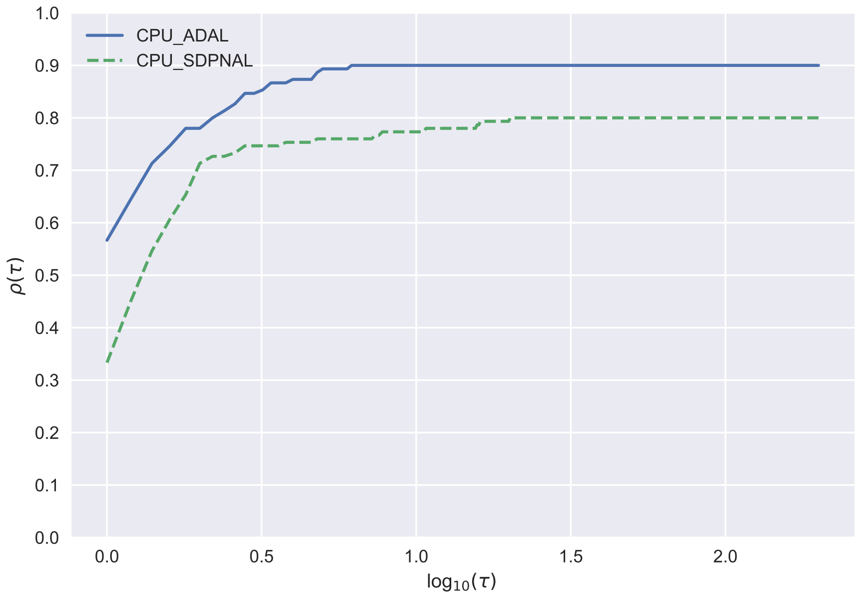


Figure 6.1: Performance profiles on CPU time. Comparison between `ADAL` and `SDPNAL+` on random instances.

As a preliminary test, we ran the version of `ADAL` tailored for SDPs in standard form. However, this non-optimized version of `ADAL` did not allow us to solve any instance due to memory issues. Therefore, in the following comparisons, we only consider the version of `ADAL` described in Section 6.1.

In Table 6.1, for each solver and each combination of n, m and p, we report the number of instances solved within the time limit and the average running time. We notice that for $n = 250$ and $m = 25000$, `SDPNAL+` is not able to solve any instance within the time limit, while `ADAL` is able to solve all of them with a precision of 10^{-5}. For $n = 500$ and $m = 100000$ both algorithms are not able to solve any instance within the time limit. `SDPNAL+` performs better on instances with $n = 1000$ and $m = 10000$, while for the other instances either the two solvers show similar performances or `ADAL` outperforms `SDPNAL+`. The performance profiles of `ADAL` and `SDPNAL+` on random instances are reported in Figure 6.1, showing the better performance of `ADAL` with respect to `SDPNAL+`: on almost 60% of the instances `ADAL` is the fastest algorithm and it is also able to solve 90% of the instances while `SDPNAL+` solves only 80% of the instances within the time limit.

6.3.2 Safebounds for the stability number and the chromatic number

The focus of this section is, from the one hand to compare the performance of `ADAL` and `SDPNAL+` on SDP relaxations for MSSP and GCP and from the other hand, to investigate the employment of the three different procedures to compute valid bounds we reported before. In particular, for the latter we are going to discuss the quality of the best safe bound computed, the time at which such

			ADAL		SDPNAL+	
n	m	p (%)	#sol	CPU time	#sol	CPU time
200	10000	25	5	39.24	5	33.05
		50	5	58.24	5	109.14
		75	5	67.14	5	713.82
250	5000	25	5	7.99	5	11.05
		50	5	9.87	5	15.51
		75	5	11.28	5	16.93
	25000	25	5	838.04	0	-
		50	5	1166.45	0	-
		75	5	1114.52	0	-
500	10000	25	5	15.52	5	15.54
		50	5	16.49	5	22.45
		75	5	28.87	5	23.94
	25000	25	5	18.11	5	31.33
		50	5	30.20	5	50.78
		75	5	45.53	5	52.57
	50000	25	5	217.61	5	106.28
		50	5	260.43	5	221.66
		75	5	325.71	5	250.97
	100000	25	0	-	0	-
		50	0	-	0	-
		75	0	-	0	-
1000	10000	25	5	136.63	5	49.52
		50	5	157.21	5	58.22
		75	5	242.63	5	71.38
	50000	25	5	57.19	5	60.96
		50	5	94.09	5	109.48
		75	5	110.00	5	111.29
	100000	25	5	83.15	5	136.53
		50	5	127.37	5	181.13
		75	5	155.05	5	184.21

Table 6.1: Results on 120 random instances

bound is found and the overhead needed to apply such procedures during the iterations of `ADAL`.

For this experiment we are going to consider the formulation of $\theta^+(G)$ based on `th-SDP1` for the stable set problem and the formulation of $\theta'(G)$ based on `th-SDP3` for the graph coloring problem. In what is following, we will refer to the matrix variables of `th-SDP1` and `th-SDP3` as Y and X, respectively. Note that in both formulations the entries of Y and X are bounded from below. In the context of ADMMs defined over the dual problem, bounds on the matrix variable can be handled by introducing a further step, where a projection onto the nonnegative orthant is performed (see e.g. [14, 77, 78]). Although these 3-blocks ADMMs may not theoretically converge [15], they perform well in practice.

While the application of the Dual bound methodology does not require additional information on the SDP, in order to use the Error bound and the Norm bound a superior limitation of the maximum eigenvalues $\lambda_{\max}(Y)$, $\lambda_{\max}(X)$ and of the norms $||Y||$, $||X||$ on optimal solutions are needed.

An exact formulation for the MSSP can be obtained by adding the rank-1 constraint to `th-SDP1`. Hence, any matrix Y that is feasible for MSSP only has entries in $\{0,1\}$. Therefore, $||Y||^2$ equals the number of nonzero entries of Y. Obviously, $||Y||$ is maximized by any optimal solution Y of MSSP. Now let Y^* denote the optimal solution with objective value α. Y^* has in total $2\alpha+1$ nonzero entries in the first row and column. Moreover, there are α^2 other nonzero entries in the remaining rows and columns. Therefore we have that

$$||Y^*||^2 = 2\alpha + 1 + \alpha^2 = (\alpha+1)^2$$

and hence, $||Y^*|| = \alpha + 1$. Thus, given any upper bound $\beta \geq \theta^+(G)$ on the stability number,

$$||Y|| \leq \beta + 1,$$

holds for any Y that is optimal for `th-SDP1`. If no better bound is known, we notice that a trivial (initial) upper bound for MSSP is given by $\beta^0 = |V|$ (which can be lowered to $\left\lfloor \frac{|V|}{2} \right\rfloor$ if the graph is connected). Then, given a dual pair (μ, Z) computed at a certain iteration of the ADMM, by applying (6.13) with $U = \beta^0$ we can compute a safe bound, say nb^0. Since the latter is a safe upper bound on the stability number, we can use nb^0 to further improve the initial estimate β^0 by the following:

$$\beta^1 = \min(\beta^0, \lfloor nb^0 \rfloor + 1),$$

which in turn will help to improve the subsequent values obtained by the Norm bound and notice that this process can be applied each time a new safe bound is computed. As regarding $\lambda_{\max}(Y)$ we can say, by (6.12):

$$\lambda_{\max}(Y) \leq \sum_i \lambda_i(Y) = \mathrm{tr}(Y) = \theta^+(G) + 1 \leq \beta^0 + 1,$$

holds for any feasible solution Y. By the same argument as before, each time time an Error bound is computed, we can improve β accordingly.

As remarked by Meurdesoif [59], the exact formulation of the GCP based on `th-SDP3` is given by a bivalent program where the entries X_{ij}'s assume values in $\{-1, t-1\}$, in particular:

$$X_{ij} = \begin{cases} t-1 & \text{if } i \text{ and } j \text{ have the same color,} \\ -1 & \text{if } i \text{ and } j \text{ have distinct colors.} \end{cases}$$

Clearly $||X||$ is minimized by the optimum X^*. Hence, a straightforward upper bound on $||X||$ can be given by a feasible coloring. Let $c : V \to 1, ..., \psi$ be a feasible coloration for the graph G with value $\psi \geq \theta'(G)$. Now, consider the following matrix

$$X^{\psi}_{ij} = \begin{cases} \psi - 1 & \text{if } c(i) = c(j) \\ -1 & \text{otherwise,} \end{cases} \tag{6.14}$$

then $||X^*|| \leq ||X^{\psi}||$. Here, since the safe bounds we are going to compute are lower estimates, we cannot use the latter to improve the upper bound on the norm. As for the stable set problem, in order to bound $\lambda_{\max}(X)$ of an optimal solution we are going to use (6.12):

$$\lambda_{\max}(X) \leq \sum_i \lambda_i(X) = \mathrm{tr}(X) = |V|(\theta'(G) - 1) \leq |V|(\psi - 1).$$

For these experiments we are going to consider the following setup: as benchmarks we are going to formulate and solve $\theta^+(G)$ and $\theta'(G)$ on graphs taken from the second DIMACS implementation challenge [39], complementing clique instances to convert them into stable set instances. Both solvers `ADAL` and `SDPNAL+` have been set with an accuracy of 10^{-6}, along with a time limit of one hour. The solver GUROBI has been set to a feasibility tolerance of 10^{-5} for the resolution of (6.10). Moreover, we equip `ADAL` with the Dual, Error and Norm bound procedures which will be applied every 200 iteration of `ADAL` and at its last, keeping the best bound found by each procedure along with the time at which it has been identified. Thus, besides the comparison between `ADAL` and `SDPNAL+` we are going to measure which procedure produces the best bounds, which is faster to detect a good bound and a comparison of the overheads yield by the application of these procedures.

Stable set instances

Table 6.2 reports results on 51 DIMACS instances. For each solver (`ADAL` and `SDPNAL+`) we reported the optimal values (ObjVal) along with the time in seconds needed to reach the optimality conditions (CPU-time). Then, for each safe bound procedure (Norm, Error and Dual Bound) we reported the best safe bound computed through the algorithm and the time needed to detect it. Notice that if `ADAL`'s CPU-time is less than some values in "found at" columns, this means that the corresponding valid bound have been identified at the very last iteration of the algorithm. While for Error and Dual Bounds we also reported the total time spent by these procedures (tot-time) within the `ADAL`'s CPU-time, we omitted the time spent by the Norm Bound since for all instances it was not superior than 0.01 seconds.

First of all, we can notice that `ADAL` and `SDPNAL+` were able to solve all instances but `p_hat1500-2`,

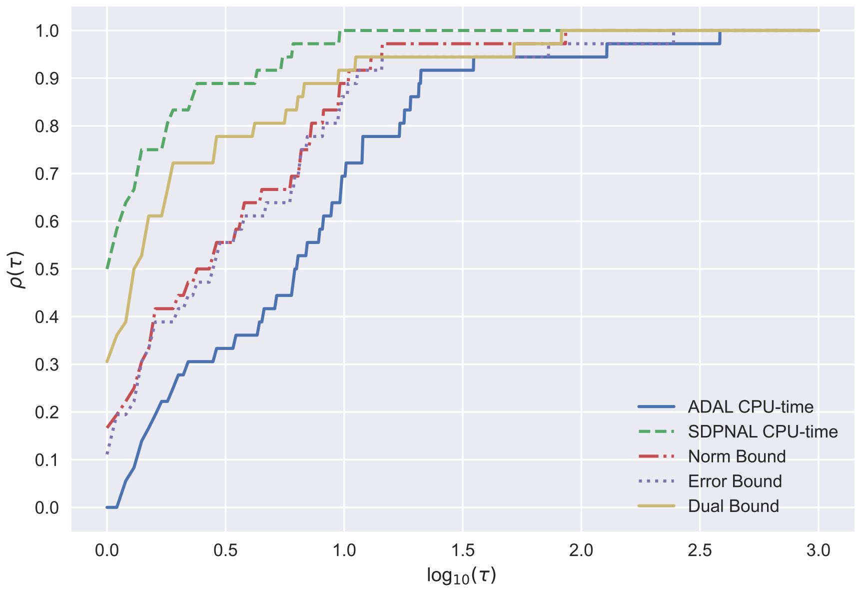


Figure 6.2: Performance profiles on CPU times for Stable Set instances.

where both solvers show a failure and `keller6`, where `ADAL` was not able to converge within the time limit. Moreover, the Norm bound, the Error bound and the Dual bound were able to identify a safe bound on all instances. In particular, we are interested in how many cases there is an integrality gap between the value of the safe bounds and the ObjVal (if available) of `SDPNAL+` (i.e. considering their rounding down to the closest integer). Hence, we report that:

- the best Norm bound differs to ObjVal in only 1 out of 51 cases;
- the best Error bound differs to ObjVal in 14 out of 50 cases;
- the best Dual bound differs to ObjVal in 0 out of 51 cases.

We remark that the difficulty of the Error bound to find a tight safe bound in these cases is due to the fact that, since this procedure is applied every 200 iterations of the algorithm, the Error bound was not capable to effectively update the upper bound on $\lambda_{\max}(Y)$ when `ADAL` converges in few hundreds iterations. In order to overcome to this phenomenon it would be sufficient to apply the Error bound more often. On the other hand, we notice for the Norm bound that even starting from a trivial upper bound on the norm $||Y||$, just few updates have been sufficient to yield a good safe bound.

In Figure 6.2, the performance profiles of `ADAL` and `SDPNAL+`'s CPU-time, along with the times to identify the best Norm, Error and Dual bound are shown. It is clear that on these instances `SDPNAL+` outperforms `ADAL`. However, we want to underline the good performances of the latter with the employment of the Dual bound, which was able to outperform `SDPNAL+` and the other safe bound procedures in about 30% of the instances, finding a tight valid bound and in particular on the `p-hat` graphs, where we are often able to get the same bound as the optimal dual objective of `SDPNAL+` in a much lower CPU time.

Clearly, solving an LP every 200 iterations has a non trivial cost for large instances (as confirmed by Dual bound's tot-time), but this overhead can be easily mitigated by seldom applying this procedure during `ADAL`'s iterations.

	`ADAL`		`SDPNAL+`		Norm Bound		Error Bound			Dual Bound		
Graph	ObjVal	CPU-time	ObjVal	CPU-time	best	found at	best	found at	tot-time	best	found at	tot-time
`DSJC125.1`	38.04	9.64	38.04	5.19	38.05	6.69	38.05	7.11	0.05	38.06	6.26	0.27
`DSJC125.5`	11.40	2.68	11.40	2.57	11.40	2.13	11.41	2.13	0.01	11.40	2.64	0.11
`DSJC125.9`	4.00	2.20	4.00	2.72	4.00	2.00	4.00	2.00	0.01	4.00	2.22	0.06
`DSJC500-5`	22.57	9.27	22.57	5.75	22.66	6.97	31.06	7.07	0.09	22.57	8.67	1.04
`DSJC1000-5`	31.67	33.85	31.67	35.80	31.93	25.75	55.76	26.19	0.43	31.67	32.62	4.56
`C125-9`	37.55	8.44	37.55	1.93	37.55	6.96	37.57	5.67	0.04	37.55	5.47	0.25
`C250-9`	55.82	36.65	55.82	3.77	55.82	27.23	55.82	26.21	0.35	55.86	21.21	1.73
`C500-9`	83.58	151.73	83.58	8.88	83.59	114.80	83.61	100.41	1.77	83.61	99.28	8.14
`C1000-9`	122.60	690.36	122.60	36.30	122.64	522.33	122.61	522.75	8.42	122.87	342.82	42.21
`C2000-5`	44.56	150.67	44.56	536.32	45.31	117.69	136.03	119.01	1.32	44.56	145.41	18.95
`C2000-9`	177.73	2836.23	177.73	280.32	177.78	2672.59	177.75	2726.72	40.40	177.95	1784.69	194.49
`brock200_1`	27.20	5.01	27.20	1.77	27.20	3.81	27.20	3.83	0.05	27.20	5.11	0.26
`brock200_2`	14.13	3.06	14.13	2.43	14.13	2.25	14.17	2.27	0.04	14.13	3.15	0.18
`brock200_3`	18.67	3.30	18.67	2.86	18.67	2.43	18.76	2.44	0.03	18.67	3.40	0.18
`brock200_4`	21.12	4.14	21.12	2.89	21.12	2.98	21.28	2.99	0.03	21.12	3.70	0.24
`brock400_1`	39.33	15.07	39.33	5.10	39.34	11.83	40.50	11.88	0.16	39.33	15.37	0.73
`brock400_2`	39.20	15.04	39.20	5.12	39.20	12.93	40.33	13.04	0.31	39.20	15.33	0.73
`brock400_3`	39.16	15.88	39.16	4.93	39.17	11.92	40.31	11.97	0.16	39.16	16.19	0.76
`brock400_4`	39.23	15.71	39.23	4.50	39.24	13.12	40.33	13.17	0.26	39.23	16.01	0.73
`brock800_1`	41.87	33.97	41.87	13.06	41.88	27.24	88.37	27.49	0.50	41.87	35.24	2.20
`brock800_2`	42.10	34.87	42.10	14.13	42.11	27.31	90.01	27.56	0.49	42.10	36.18	2.23
`brock800_3`	41.88	33.46	41.88	14.65	41.89	27.75	84.12	27.99	0.48	41.88	34.74	2.19
`brock800_4`	42.00	34.81	42.00	14.00	42.01	27.33	86.63	27.57	0.50	42.00	36.11	2.25
`p_hat300-1`	10.02	58.09	10.02	17.33	10.02	14.37	10.02	14.39	0.13	10.02	9.17	3.40
`p_hat300-2`	26.71	1058.64	26.71	163.11	26.71	196.52	26.71	196.55	1.75	26.71	30.17	60.69
`p_hat300-3`	40.70	144.44	40.70	37.10	40.70	56.60	40.70	56.63	0.57	40.70	20.84	7.85
`p_hat500-1`	13.01	142.97	13.01	16.13	13.01	37.56	13.02	37.64	0.48	13.01	24.06	8.79
`p_hat500-2`	38.56	725.88	38.56	538.33	38.56	723.20	38.56	723.29	8.62	38.56	88.84	208.43
`p_hat500-3`	57.81	639.90	57.81	35.68	57.81	210.51	57.81	210.61	2.62	57.81	65.02	36.70
`p_hat700-1`	15.05	282.60	15.05	35.87	15.05	71.31	15.07	71.50	0.82	15.05	61.08	17.48
`p_hat700-2`	48.44	1788.07	48.44	297.81	48.44	1787.44	48.44	1776.18	18.61	48.44	170.97	217.28
`p_hat700-3`	71.76	1961.95	71.76	95.01	71.76	615.68	71.76	615.85	7.61	71.76	141.00	116.32
`p_hat1000-1`	17.52	404.67	17.52	120.12	17.52	115.77	17.57	116.09	1.61	17.52	88.78	27.78
`p_hat1000-2`	54.84	2852.66	54.84	698.39	54.84	2851.17	54.85	2851.48	31.70	54.85	302.03	245.96
`p_hat1000-3`	83.53	2337.28	83.53	244.45	83.53	694.69	83.54	695.04	8.38	83.53	242.76	154.07
`p_hat1500-1`	21.89	1118.44	21.89	480.53	21.89	295.59	21.99	296.27	3.45	21.89	216.50	79.24
`p_hat1500-2`	-	-	-	-	76.46	3565.92	76.48	3494.53	34.54	76.46	872.20	234.78
`p_hat1500-3`	113.65	3014.54	113.65	881.41	113.66	3014.01	113.66	3014.70	35.88	113.65	958.14	226.53
`keller4`	13.47	16.44	13.47	2.74	13.47	10.17	13.47	10.18	0.09	13.47	5.18	0.86
`keller5`	31.00	1281.91	31.00	54.41	31.00	499.35	31.62	499.55	6.73	31.00	755.56	67.24
`keller6`	-	-	63.00	1526.56	63.03	1535.40	288.88	1541.51	20.91	63.00	1913.78	79.56
`sanr200_0.7`	23.63	4.81	23.63	2.19	23.64	3.30	23.72	3.32	0.03	23.63	3.76	0.28
`sanr200_0.9`	48.90	20.16	48.90	3.29	48.91	14.49	48.91	15.14	0.17	48.91	13.69	1.02
`sanr400_0.5`	20.18	6.98	20.18	4.45	20.25	4.83	23.90	4.88	0.05	20.18	7.27	0.52
`sanr400_0.7`	33.97	10.54	33.97	6.01	33.97	8.81	34.88	8.91	0.15	33.97	9.82	0.79
`MANN_a9`	17.48	0.61	17.48	1.61	17.48	0.56	17.48	0.54	0.00	17.48	0.38	0.05
`MANN_a27`	132.76	838.69	132.76	6.55	132.76	561.87	132.77	475.86	8.62	132.94	340.53	43.47
`hamming6-2`	32.00	11.54	32.00	1.29	32.75	0.97	32.00	5.95	0.03	32.00	6.54	0.50
`hamming6-4`	4.00	0.20	4.00	1.62	4.01	0.17	4.07	0.17	0.00	4.00	0.21	0.01
`hamming8-2`	128.00	1951.74	128.00	5.09	128.53	36.17	128.00	1245.57	23.90	128.00	419.50	104.14
`hamming8-4`	16.00	3.78	16.00	2.79	16.00	2.84	16.00	2.87	0.07	16.00	2.38	0.27
`hamming10-4`	42.67	60.55	42.67	32.69	42.68	50.41	42.76	31.87	2.99	42.76	33.11	4.28

Table 6.2: Safe bounds of the stability number on DIMACS instances.

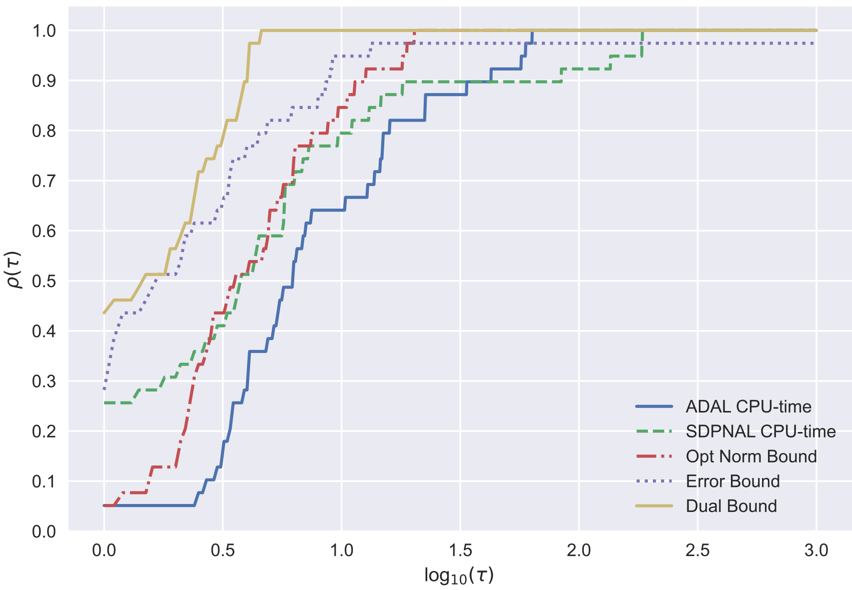


Figure 6.3: Performance profiles on CPU times for Graph Coloring instances.

Graph coloring instances

In Tables 6.3 and 6.4 results on 112 DIMACS instances have been reported. As before, for each solver (`ADAL` and `SDPNAL+`) we reported the optimal values (ObjVal) along with the time in seconds needed to reach the optimality conditions (CPU-time). Then, for each safe bound procedure (Norm, Error and Dual Bound) we reported the best safe bound computed through the algorithm and the time needed to detect it. Again, while for Error and Dual Bounds we also reported the total time spent by these procedures (tot-time) within the `ADAL`'s CPU-time, we omitted the time spent by the Norm Bound since for all instances it was not superior than 0.01 seconds. Moreover, since in this experiments we noticed that the upper bound on the norm $||X||$ we proposed in (6.14), i.e. computed by $||X^{\psi}||$ yield by a feasible $\psi - coloring$ was too large w.r.t. the optimal norm $||X^*||$, we opted to report an additional column as a reference, namely "Opt Norm Bound" where this procedure has been applied by using $U = ||X^*||$, where X^* is an optimal solution for $\theta'(G)$ computed beforehand, so that the potential of this method can be shown.

As we can notice, `SDPNAL+` have been able to solve all instances to optimality within the time limit, while `ADAL` have shown failure in 5 instances. On the other hand, `ADAL` outperforms `SDPNAL+` in 35 out of 107 solved instances.

In this experiment the Norm and the Dual bound were able to identify a valid bound in, respectively, 92 and 94 out of 112 cases, while the Error and the Norm bound with the optimal U were always able to find a safe bound. As for the stable set problem, it is of interest to consider the cases in which there is integrality gap between the objective value (ObjVal) of the solver `SDPNAL+` and the valid bounds (here, since we are minimizing the rounding up to the closest integer must be considered). In particular, we have:

- the best Norm bound differs to ObjVal in 45 out of 92 cases;
- the best Error bound differs to ObjVal in only 4 out of 112 cases;
- the best Dual bound differs to ObjVal in 2 out of 94 cases;
- the best Opt Norm bound differs to ObjVal in 0 out of 112 cases.

From these data we can clearly point out that the Norm bound procedure relies on a "good" estimate of U. As a matter of fact, we experienced that $||X^{\psi}||$ could be up to 3 orders of magnitude bigger than the actual optimal $||X^{*}||$, moreover the impossibility of an update of U during the iterations of `ADAL`, as we did for the stable set instances, does not turn in favor of this method. On the contrary, the estimate on $\lambda_{\max}(X)$ gave the possibility to identify a valid bound in all cases which is almost always close to the optimal value of the solvers.

The effectiveness of the Dual bound method is also reflected in the performance profiles, as shown if Figure 6.3. In the performance profiles, we excluded those instances for which the difference in absolute value of the best valid bounds found by Opt Norm, Error and Dual bound methods and the objective of `SDPNAL+` is less than 0.5. In particular, we excluded all the instances where the post-processing procedure was not able to compute a bound. The Dual bound was the first able to identify the best valid bound in almost 45% of the instances, while the Error bound outperforms both the solvers and the other procedures in almost 30% of the cases. Again, we remark that for large instances the computational burden of the Dual bound is nontrivial, but it can be lowered by changing the frequency of the calls to the procedure.

As a further comparison between `ADAL` and `SDPNAL+` on SDP relaxations of the chromatic number, we built instances adding 1000, 2500 and 5000 inequalities to `th-SDP3` for $\vartheta(G)$. Such inequalities have been randomly selected among the so-called triangle inequalities proposed by Meurdesoif in [59].

Results are reported in Table 6.5. Among 45 total SDPs `SDPNAL+` was able to converge to optimality within 1 hour in 35 cases, while `ADAL` solves 36 instances. Moreover, we can identify 12 instances where the CPU-time of `ADAL` is lower than `SDPNAL+`'s, in particular the first performs remarkably well on most of `DSJC` instances and `abb313GPIA`.

Similarly as before, these results confirm the need of a good estimate U to employ the Norm bound procedure, while Error and Dual bound were able to find a tight valid lower bound in a time much less than the convergence time of `ADAL`.

	ADAL		SDPNAL+		Norm Bound		Opt Norm Bound		Error Bound			Dual Bound		
Graph	ObjVal	CPU-time	ObjVal	CPU-time	best	found at	best	found at	best	found at	tot-time	best	found at	tot-time
DSJC125.1	4.14	69.01	4.14	22.88	3.08	3.55	4.14	8.39	4.12	3.55	0.03	4.14	1.72	0.91
DSJC125.5	11.87	1.02	11.87	1.36	11.61	0.32	11.87	0.69	11.86	0.32	< 0.01	11.87	0.74	0.06
DSJC125.9	37.80	1.87	37.80	2.03	37.76	0.54	37.80	1.23	37.79	0.54	0.01	37.80	1.31	0.09
DSJC250.1	4.94	6.65	4.94	6.70	1.86	1.32	4.94	2.85	4.92	1.33	0.03	4.94	2.19	0.16
DSJC250.5	16.35	1.90	16.35	2.84	15.73	0.69	16.35	1.62	16.33	0.70	0.01	16.35	2.00	0.10
DSJC250.9	55.22	4.76	55.22	3.76	55.06	1.58	55.22	3.85	55.20	1.59	0.04	55.22	3.76	0.37
DSJC500.1	6.25	8.66	6.25	16.57	1.59	3.75	6.25	6.17	6.22	3.79	0.10	6.25	5.91	0.16
DSJC500.5	22.90	5.41	22.90	9.64	18.90	2.27	22.90	4.03	22.81	2.31	0.05	22.90	5.71	0.30
DSJC500.9	84.14	17.04	84.14	16.36	80.85	6.36	84.14	12.54	83.03	6.36	0.02	84.14	17.53	1.21
DSJR500.1	12.00	35.18	12.00	10.00	10.09	13.53	12.00	21.92	12.00	15.04	0.03	12.00	23.59	0.34
DSJR500.1c	-	-	83.75	1231.74	83.45	506.26	83.74	941.95	83.69	506.27	1.17	83.75	190.31	294.26
DSJR500.5	122.01	198.08	122.00	16.95	108.36	52.64	122.00	85.38	122.00	52.65	0.29	122.00	181.80	6.48
DSJC1000.1	8.36	31.65	8.36	59.76	-	-	8.36	20.30	8.23	15.27	0.03	8.36	22.83	0.57
DSJC1000.5	32.11	18.30	32.11	38.11	13.94	10.39	32.11	14.32	31.82	10.59	0.20	32.11	19.46	1.17
DSJC1000.9	122.80	72.30	122.80	63.88	90.40	37.11	122.79	54.62	122.80	37.14	0.09	122.80	70.85	6.89
fpsol2.i.1	65.00	200.57	65.00	11.71	-	-	65.00	139.90	64.97	87.18	0.17	65.00	199.60	2.67
fpsol2.i.2	30.00	28.11	30.00	9.75	18.69	14.79	30.00	22.71	30.00	14.79	0.03	30.00	25.71	0.43
fpsol2.i.3	30.00	27.36	30.00	7.82	19.53	13.42	30.00	18.77	30.00	13.42	0.03	30.00	27.43	0.42
inithx.i.1	54.00	604.51	54.00	32.25	-	-	54.00	339.96	53.99	260.84	0.47	54.00	537.71	4.96
inithx.i.2	31.00	387.80	31.00	12.39	29.57	23.17	30.98	230.08	30.99	159.76	0.27	30.22	35.60	3.70
inithx.i.3	31.00	341.12	31.00	13.64	29.64	21.09	31.00	152.15	31.00	102.87	0.23	30.23	32.57	3.45
latin_square_10	90.00	48.40	89.99	41.12	-	-	90.00	40.18	89.87	19.98	< 0.01	-	-	2.91
le450_15a	15.00	6.37	15.00	5.18	-	-	15.00	5.13	15.00	3.31	< 0.01	-	-	0.12
le450_15b	15.00	7.06	15.00	5.61	-	-	15.00	5.04	15.00	3.25	< 0.01	15.00	7.13	0.14
le450_15c	15.00	3.92	15.00	4.70	6.62	1.78	15.00	2.70	15.00	1.78	< 0.01	15.00	4.02	0.10
le450_15d	15.00	3.86	15.00	4.71	6.59	1.75	15.00	2.88	15.00	1.76	< 0.01	15.00	3.96	0.10
le450_25a	25.00	19.73	25.00	7.54	-	-	25.00	13.61	24.99	9.23	< 0.01	25.00	19.53	0.29
le450_25b	25.00	18.44	25.00	7.27	21.53	9.06	25.00	13.45	24.99	9.23	< 0.01	-	-	0.23
le450_25c	25.00	9.67	25.00	7.03	-	-	25.00	6.88	24.98	3.04	< 0.01	25.00	9.78	0.20
le450_25d	25.00	9.15	25.00	6.82	7.89	4.74	25.00	7.12	25.00	4.75	< 0.01	25.00	9.25	0.19
mulsol.i.1	49.00	18.89	49.00	3.48	30.75	5.52	49.00	9.97	48.99	5.52	0.01	-	-	0.66
mulsol.i.2	31.00	9.02	31.00	2.92	25.69	2.95	31.00	7.10	31.00	2.89	0.01	31.00	9.04	0.28
mulsol.i.3	31.00	8.13	31.00	3.12	25.14	2.66	30.99	6.94	31.00	2.66	0.01	31.00	7.47	0.19
mulsol.i.4	31.00	7.97	31.00	2.40	25.03	2.73	30.99	7.03	31.00	2.73	< 0.01	31.00	6.31	0.22
mulsol.i.5	31.00	9.88	31.00	3.34	26.78	3.31	31.00	8.41	31.00	3.57	0.01	31.00	9.01	0.19
school1	14.00	14.74	14.00	65.03	10.69	5.87	14.00	10.52	13.99	5.87	0.01	14.00	8.08	0.40
school1_nsh	14.00	12.12	14.00	75.97	9.40	3.94	14.00	7.97	13.98	3.96	< 0.01	14.00	7.29	0.30
zeroin.i.1	49.00	24.91	49.00	2.47	25.89	8.04	49.00	16.68	48.94	6.43	0.02	49.00	21.91	0.80
zeroin.i.2	30.00	14.63	30.00	2.43	19.63	5.15	30.00	13.74	30.00	5.02	0.01	30.00	14.13	0.48
zeroin.i.3	30.00	14.62	30.00	2.80	18.98	4.97	29.99	7.17	30.00	4.97	0.01	30.00	13.53	0.46
anna	11.00	9.97	11.00	1.16	7.56	3.18	11.00	6.24	11.00	3.18	0.01	-	-	0.13
david	11.00	2.46	11.00	0.59	10.82	0.76	11.00	1.78	11.00	0.76	< 0.01	-	-	0.08
huck	11.00	1.60	11.00	0.43	10.91	0.47	11.00	1.13	11.00	0.49	< 0.01	-	-	0.04
jean	10.00	1.35	10.00	0.53	9.18	0.39	10.00	0.90	10.00	0.33	< 0.01	-	-	0.03
games120	9.00	3.23	9.00	0.86	7.20	0.94	9.00	1.69	9.00	0.94	< 0.01	-	-	0.07
miles250	8.00	7.17	8.00	0.94	6.20	2.10	8.00	4.85	8.00	2.10	< 0.01	8.00	6.30	0.11
miles500	20.00	6.78	20.00	1.69	15.13	1.98	20.00	4.52	19.99	1.35	< 0.01	20.00	6.26	0.11
miles750	31.00	4.75	31.00	2.73	28.51	1.52	31.00	3.57	30.75	0.54	0.01	31.00	4.77	0.09
miles1000	42.00	7.81	42.00	1.64	39.97	1.37	42.00	4.87	41.97	1.93	< 0.01	42.00	7.61	0.16
miles1500	73.00	10.36	73.00	1.48	70.02	2.75	73.00	6.66	72.87	2.00	0.01	73.00	10.28	0.27
queen5_5	5.00	0.01	5.00	0.11	4.99	0.01	5.00	0.01	5.00	0.01	< 0.01	5.00	0.04	0.03
queen6_6	6.04	0.77	6.04	0.69	6.04	0.21	6.04	0.29	6.04	0.21	< 0.01	6.04	0.19	0.04
queen7_7	7.00	0.08	7.00	0.29	6.94	0.03	7.00	0.06	7.00	0.03	< 0.01	7.00	0.08	0.01
queen8_8	8.00	0.10	8.00	0.19	7.82	0.03	8.00	0.08	7.97	0.03	< 0.01	8.00	0.11	0.01
queen8_12	12.00	0.55	12.00	0.62	11.26	0.19	12.00	0.42	11.90	0.19	< 0.01	-	-	0.02
queen9_9	9.00	0.15	9.00	0.23	8.36	0.05	9.00	0.12	8.91	0.05	< 0.01	9.00	0.16	0.01
queen10_10	10.00	0.23	10.00	0.44	9.18	0.08	10.00	0.18	10.00	0.08	< 0.01	10.00	0.24	0.01
queen11_11	11.00	0.46	11.00	0.47	10.12	0.13	11.00	0.29	10.90	0.13	< 0.01	11.00	0.47	0.01
queen12_12	12.00	0.67	12.00	0.68	7.97	0.17	12.00	0.37	12.00	0.17	< 0.01	12.00	0.71	0.04
queen13_13	13.00	0.76	13.00	0.64	8.18	0.25	13.00	0.66	12.56	0.25	< 0.01	13.00	0.80	0.04
queen14_14	14.00	1.27	14.00	0.82	11.35	0.37	14.00	0.76	13.78	0.37	< 0.01	14.00	1.32	0.04
queen15_15	15.00	1.38	15.00	1.26	6.25	0.44	15.00	1.01	15.00	0.44	< 0.01	-	-	0.04
queen16_16	16.00	1.84	16.00	1.46	7.11	0.61	16.00	1.57	15.36	0.62	< 0.01	16.00	1.90	0.06
myciel3	2.40	0.01	2.40	0.13	2.40	0.00	2.40	0.01	2.40	0.00	< 0.01	2.40	0.04	0.03
myciel4	2.53	0.04	2.53	0.18	2.53	0.01	2.53	0.02	2.53	0.01	< 0.01	2.53	0.04	0.01
myciel5	2.64	0.41	2.64	0.41	2.64	0.06	2.64	0.21	2.64	0.06	< 0.01	2.64	0.23	0.02
myciel6	2.73	1.73	2.73	1.16	2.73	0.36	2.73	0.81	2.73	0.36	< 0.01	2.73	0.51	0.04
myciel7	2.82	7.35	2.82	7.60	2.79	1.99	2.82	4.47	2.82	1.99	< 0.01	2.82	1.34	0.24
mug88_1	3.00	11.78	3.00	29.45	2.65	0.72	3.00	1.62	3.00	0.72	< 0.01	3.00	0.35	0.24
mug88_25	3.00	20.81	3.00	47.43	2.73	1.03	3.00	2.20	3.00	1.03	< 0.01	3.00	0.35	0.43
mug100_1	3.00	19.59	3.00	84.51	2.61	0.97	3.00	2.93	3.00	0.97	< 0.01	3.00	0.46	0.31
mug100_25	3.00	26.20	3.00	84.97	2.69	2.04	3.00	4.42	3.00	2.04	< 0.01	3.00	0.46	0.39

Table 6.3: Safe bounds of the chromatic number on DIMACS instances.

	ADAL		SDPNAL+		Norm Bound		Opt Norm Bound		Error Bound			Dual Bound		
Graph	ObjVal	CPU-time	ObjVal	CPU-time	best	found at	best	found at	best	found at	tot-time	best	found at	tot-time
abb313GPIA	8.00	615.04	8.00	2949.22	-	-	8.00	317.51	7.84	245.06	0.46	8.01	55.61	4.48
ash331GPIA	3.38	125.34	3.38	17.85	3.12	60.83	3.38	88.25	3.37	60.85	0.12	3.38	38.24	1.01
ash608GPIA	3.33	265.72	3.33	41.34	2.84	92.45	3.33	201.23	3.33	138.20	0.25	3.31	129.25	1.34
ash958GPIA	3.33	529.68	3.33	124.35	-	-	3.33	412.24	3.19	197.96	1.13	-	-	2.51
will199GPIA	6.10	156.39	6.10	32.11	4.05	90.26	6.10	132.49	6.09	90.28	0.25	6.10	124.61	1.22
1-Insertions_4	2.23	1.93	2.23	1.01	2.23	0.40	2.23	0.98	2.23	0.40	< 0.01	2.23	0.34	0.07
1-Insertions_5	2.28	19.71	2.28	15.04	2.26	5.62	2.28	12.96	2.28	5.62	< 0.01	2.28	2.64	0.56
1-Insertions_6	2.31	337.22	2.31	100.65	2.23	140.52	2.31	200.17	2.31	140.53	0.2	2.31	22.80	3.29
2-Insertions_3	2.10	0.38	2.10	0.57	2.10	0.10	2.10	0.11	2.10	0.10	< 0.01	2.10	0.18	0.04
2-Insertions_4	2.13	25.27	2.13	9.06	2.13	5.31	2.13	11.76	2.13	5.31	< 0.01	2.13	1.59	0.36
2-Insertions_5	2.16	544.91	2.16	109.90	2.11	207.86	2.16	295.30	2.16	207.87	0.4	2.16	52.67	4.99
3-Insertions_3	2.07	1.22	2.07	1.00	2.06	0.23	2.07	0.55	2.07	0.23	< 0.01	2.07	0.31	0.06
3-Insertions_4	2.09	125.48	2.09	29.79	2.08	26.00	2.09	52.68	2.09	26.01	< 0.01	2.09	8.39	2.37
3-Insertions_5	-	-	2.10	3568.47	1.92	2551.74	2.10	3255.45	2.10	2551.80	4.8	2.11	130.38	17.94
4-Insertions_3	2.05	2.39	2.05	2.75	2.03	0.40	2.05	1.23	2.05	0.40	< 0.01	2.05	0.38	0.08
4-Insertions_4	2.06	563.58	2.06	130.23	2.04	118.00	2.06	160.38	2.06	118.01	0.2	2.06	8.89	6.64
1-FullIns_3	3.06	0.32	3.06	0.35	3.06	0.14	3.06	0.20	3.06	0.14	< 0.01	3.06	0.13	0.05
1-FullIns_4	3.12	4.37	3.12	2.45	3.12	1.48	3.12	3.31	3.12	1.48	< 0.01	3.12	1.39	0.10
1-FullIns_5	3.18	71.27	3.18	17.55	3.16	26.45	3.18	49.18	3.18	26.45	< 0.01	3.18	18.65	1.52
2-FullIns_3	4.03	1.34	4.03	0.39	4.03	0.29	4.03	1.07	4.03	0.41	< 0.01	4.03	0.74	0.08
2-FullIns_4	4.06	57.51	4.06	9.21	4.04	5.99	4.06	49.66	4.06	21.58	< 0.01	4.06	26.76	1.64
2-FullIns_5	4.08	2670.31	4.08	184.26	3.78	1461.35	4.08	1935.61	4.08	1461.37	2.8	4.08	381.59	19.29
3-FullIns_3	5.02	6.12	5.02	1.18	5.02	1.96	5.02	4.79	5.02	1.96	< 0.01	5.02	4.47	0.15
3-FullIns_4	5.03	329.51	5.03	24.03	5.03	110.09	5.03	273.46	5.03	205.20	0.3	5.03	58.31	4.94
3-FullIns_5	-	-	5.05	1769.01	1.25	3413.16	5.04	3600.31	5.03	3554.98	10.9	5.04	2965.54	18.40
4-FullIns_3	6.01	21.19	6.01	2.30	6.01	7.97	6.01	20.86	6.01	7.97	< 0.01	6.01	1.65	0.32
4-FullIns_4	6.02	1979.86	6.02	88.40	6.02	808.21	6.02	1664.70	6.02	808.22	2.3	6.02	283.54	16.15
5-FullIns_3	7.01	61.22	7.01	2.72	7.01	24.23	7.01	55.09	7.01	24.23	< 0.01	7.00	12.48	0.71
5-FullIns_4	-	-	7.01	207.83	7.01	3600.07	7.01	3600.02	7.01	3600.10	6.9	7.01	137.49	21.11
wap01a	-	-	41.00	309.86	40.61	3600.31	40.40	3600.41	40.65	3600.60	10.68	40.38	3575.61	20.34
wap02a	40.00	538.62	40.00	473.42	-	-	40.00	424.41	38.50	106.12	0.90	-	-	3.29
wap03a	40.00	1594.69	40.00	2668.31	-	-	40.00	1341.59	39.46	817.83	2.26	40.00	1507.80	10.12
wap04a	40.00	2175.13	40.00	2658.54	-	-	40.00	1948.48	40.00	1860.39	3.51	40.00	2179.94	13.84
wap05a	50.00	1099.11	50.00	24.19	-	-	50.00	609.10	50.00	621.27	1.44	50.00	918.93	11.72
wap06a	40.00	63.35	40.00	69.47	-	-	40.00	50.88	33.83	50.57	0.10	-	-	0.74
wap07a	40.00	309.93	40.00	426.97	-	-	39.98	204.64	39.36	206.91	1.84	40.00	145.89	3.00
wap08a	40.00	278.67	40.00	224.35	-	-	40.00	211.24	39.97	160.02	1.01	-	-	2.26
qg.order30	30.00	32.22	30.00	21.30	-	-	29.98	27.02	29.99	26.04	0.06	-	-	0.30
qg.order40	40.00	153.25	40.00	82.68	-	-	40.00	108.10	39.99	125.77	0.72	-	-	1.08
qg.order60	60.00	1684.60	60.00	496.86	-	-	59.99	1149.90	59.96	1232.78	3.66	-	-	9.74

Table 6.4: Safe bounds of the chromatic number on DIMACS instances.

	ADAL		SDPNAL+		Norm Bound		Opt Norm Bound		Error Bound			Dual Bound		
Graph	ObjVal	CPU-time	ObjVal	CPU-time	best	found at	best	found at	best	found at	tot-time	best	found at	tot-time
$\vartheta(G)$ + 1000 inequalities from [59]														
DSJC500.5	22.90	16.57	22.90	38.73	21.88	13.83	22.90	13.71	22.86	13.78	0.08	22.90	13.77	0.54
DSJC1000.1	8.36	34.73	8.36	154.79	-	-	8.35	24.06	8.23	24.11	0.05	8.36	25.85	0.56
DSJC1000.5	32.11	33.63	32.11	154.82	25.35	27.76	32.11	27.05	32.02	27.10	0.05	32.11	34.50	1.18
myciel7	2.85	256.14	2.85	13.77	2.85	113.71	2.85	112.93	2.85	112.93	0.14	2.85	13.39	6.77
mug88_25	3.00	17.66	3.00	33.92	2.78	2.47	3.00	3.07	2.98	3.07	0.00	-	-	0.25
mug100_25	3.00	23.13	3.00	97.57	2.74	6.55	3.00	6.61	2.99	6.61	0.03	-	-	0.28
abb313GPIA	8.00	663.11	8.00	3466.57	-	-	7.96	336.59	7.84	337.24	4.21	8.00	178.77	4.49
1-Insertions_6	2.33	1014.77	2.33	233.15	2.30	529.41	2.33	533.20	2.33	533.22	1.06	2.33	65.62	8.98
2-Insertions_5	2.18	820.34	2.18	373.95	2.14	426.93	2.18	426.83	2.18	426.84	0.77	2.18	100.30	7.21
3-Insertions_5	-	-	-	-	1.86	3600.16	2.11	3600.08	2.11	3600.17	7.77	2.11	415.32	18.32
4-Insertions_4	2.07	622.86	2.07	363.73	2.06	339.75	2.07	324.79	2.07	324.80	1.09	2.07	87.32	6.53
1-FullIns_5	3.19	2005.78	3.19	141.02	3.15	308.58	3.19	303.53	3.19	303.54	0.49	3.19	59.14	35.67
5-FullIns_4	-	-	7.01	257.83	7.01	3197.10	7.01	3210.44	7.01	3210.48	7.23	7.01	132.12	21.35
wap03a	40.00	1635.81	-	-	-	-	39.98	1374.89	39.99	1376.46	3.01	40.00	1496.25	11.16
wap04a	40.00	2300.91	-	-	-	-	39.98	1980.45	38.97	994.79	3.66	40.00	2302.30	13.66
$\vartheta(G)$ + 2500 inequalities from [59]														
DSJC500.5	22.90	80.22	22.90	43.81	21.88	13.83	22.90	13.71	22.86	13.78	0.08	22.90	79.92	0.60
DSJC1000.1	8.36	40.43	8.36	167.61	-	-	8.35	24.06	8.23	24.11	0.05	8.36	29.44	0.57
DSJC1000.5	32.11	125.35	32.11	157.33	25.35	27.76	32.11	27.05	32.02	27.10	0.05	32.11	91.06	2.39
myciel7	2.87	322.10	2.87	13.80	2.85	113.71	2.85	112.93	2.85	112.93	0.14	2.87	63.14	3.51
mug88_25	3.00	83.45	3.00	81.22	2.78	2.47	3.00	3.07	2.98	3.07	0.00	3.00	36.93	1.22
mug100_25	3.00	92.20	3.00	97.93	2.74	6.55	3.00	6.61	2.99	6.61	0.03	3.00	85.68	1.43
abb313GPIA	8.00	684.48	8.00	2611.71	-	-	7.96	336.59	7.84	337.24	4.21	8.00	250.42	4.60
1-Insertions_6	2.34	828.31	2.34	553.82	2.30	529.41	2.33	533.20	2.33	533.22	1.06	2.34	111.23	6.91
2-Insertions_5	2.19	795.11	2.19	576.61	2.14	426.93	2.18	426.83	2.18	426.84	0.77	2.19	157.16	5.79
3-Insertions_5	-	-	-	-	1.86	3600.16	2.11	3600.08	2.11	3600.17	7.77	2.12	1133.88	18.54
4-Insertions_4	2.08	619.78	2.08	537.75	2.06	339.75	2.07	324.79	2.07	324.80	1.09	2.08	143.96	4.82
1-FullIns_5	-	-	3.19	145.95	3.15	308.58	3.19	303.53	3.19	303.54	0.49	3.19	270.64	34.75
5-FullIns_4	-	-	7.01	308.79	7.01	3197.10	7.01	3210.44	7.01	3210.48	7.23	7.01	149.04	20.10
wap03a	40.00	1922.71	-	-	-	-	39.98	1374.89	39.99	1376.46	3.01	40.00	1935.99	10.83
wap04a	40.00	2630.68	-	-	-	-	39.98	1980.45	38.97	994.79	3.66	40.00	2011.30	14.42
$\vartheta(G)$ + 5000 inequalities from [59]														
DSJC500.5	22.90	462.24	22.90	46.24	21.88	13.74	22.90	13.49	22.86	13.61	0.12	22.90	456.20	1.14
DSJC1000.1	8.36	62.30	8.36	163.05	-	-	8.35	23.90	8.23	23.94	0.04	8.36	48.59	0.70
DSJC1000.5	32.11	590.96	32.11	168.79	25.35	26.53	32.11	27.07	32.02	27.11	0.04	32.11	590.91	2.12
myciel7	2.89	584.85	2.89	15.44	2.85	113.86	2.85	115.04	2.85	115.04	0.14	2.89	132.16	3.90
mug88_25	3.00	199.54	3.00	116.73	2.78	2.55	3.00	2.45	2.98	2.45	0.00	3.00	78.35	1.56
mug100_25	3.00	216.11	3.00	154.84	2.74	6.73	3.00	6.88	2.99	6.88	0.03	3.00	190.52	1.39
abb313GPIA	8.00	734.66	-	-	-	-	7.96	333.23	7.84	333.78	4.00	8.00	356.07	4.36
1-Insertions_6	2.36	1351.31	2.36	258.02	2.30	530.00	2.33	529.56	2.33	529.58	1.05	2.36	218.89	8.25
2-Insertions_5	2.19	906.51	2.19	572.94	2.14	424.90	2.18	424.94	2.18	424.96	0.77	2.19	274.81	5.39
3-Insertions_5	-	-	-	-	1.86	3600.04	2.11	3600.02	2.11	3600.11	7.72	2.12	1737.21	15.49
4-Insertions_4	2.09	756.11	2.09	831.07	2.06	317.93	2.07	322.20	2.07	322.21	1.08	2.09	315.23	4.01
1-FullIns_5	-	-	3.19	179.80	3.15	305.48	3.19	295.42	3.19	295.42	0.46	3.19	554.77	22.29
5-FullIns_4	-	-	7.01	344.28	7.01	3203.17	7.01	3198.19	7.01	3198.25	7.28	7.01	293.47	17.54
wap03a	40.00	2587.76	-	-	-	-	39.98	1452.74	39.99	1454.26	3.01	40.00	2383.76	15.00
wap04a	-	-	-	-	-	-	39.98	1987.05	38.97	1000.33	3.69	40.00	3129.61	18.72

Table 6.5: Safe bounds of the chromatic number on DIMACS instances.

Conclusions and outlook

This thesis covers Semidefinite relaxations obtained by the strong Lift-and-Project operator introduced by Lovász and Schrijver applied to two fundamental Combinatorial Optimization problems, namely the Maximum Stable Set and the Graph Coloring Problem.

In Chapter 4 we introduced a new SDP relaxation for the Stable Set problem obtained by the lifting of the so-called nodal polytope, a compact linear formulation of this problem. At first, theoretical aspects have been addressed by showing some classes of facet-defining inequalities for the stable set polytope implied by our proposal, allowing to define a clear comparison with the well-known hierarchy of SDP relaxations for this problem. Then, computational experiments have been reported. Although the application of the operator produces SDPs of polynomial size, they are challenging to solve with general-purpose SDP solver, such as interior-point methods for SDP. Hence, in order to mitigate such computational burden a cutting-plane scheme within a state-of-the-art Alternating Direction Method of Multiplier have been employed. This setup led us to investigate the behaviour of SDP relaxations on a wide benchmark.

In order to write the nodal inequality corresponding a node v in the graph G, one needs to compute the stability number $\alpha(G[\Gamma(v)])$ of the subgraph induced by its neighborhood. Clearly, replacing this coefficient with a polynomially computable upper bound, maintains the validity of the inequality for the stable set polytope. Although we were able to afford this computation in our experiments, it would be interesting to investigate the substitution of $\alpha(G[\Gamma(v)])$ with $\theta(G[\Gamma(v)])$ and studying if and how the theoretical properties we were able to prove for $N_+(\mathrm{NOD}(G))$ are affected. In the best-case scenario, this would imply the existence of a polynomial-time separation algorithm for a class of inequalities which includes all clique, orthonormal representation, odd wheel, odd antihole and almost all antiweb inequalities.

Despite Lift-and-Project applications have been widely studied on the Stable Set problem, to the best of our knowledge none on the Graph Coloring problem have been presented. Hence, in Chapter 5 we investigated the application of the Lovász and Schrijver operator to the compact representative formulation due to Campelo et al. [12]. The numerical results pointed out the existence of a class of graphs with a significant gap $\chi(G)-\theta(\bar{G})$, where the proposed SDP relaxation can yield strong lower bounds to the chromatic number. In particular, these can be better than the fractional chromatic number, which is a remarkable threshold to cross even by strong SDP relaxations. Computations have been carried out by a standard general-purpose SDP algorithm which has been able to handle only a restricted set of small graphs. To our knowledge, a deeper study of alternative relaxations of $M_+(\mathrm{REP}(G))$ may both lead to better bounds and smaller models. While, as in related experiences, designing tailored algorithms with clever symmetry handling seems the natural direction to be followed in order to increase problem size.

Although interior-point methods are well-established for small and medium size SDPs, they become soon impractical for large scale semidefinite programs, such as those obtained by the application of the Lift-and-Project operator previously discussed, for example. On the other hand, Alternating Direction Methods of Multipliers currently represent the most popular first-order alternatives, being capable to scale to much larger semidefinite programs. This of course at some cost in accuracy, that should be correctly addressed when bounding the optimal solution of some combinatorial optimization problem. To this purpose, in Chapter 6 we presented an implementation of an ADMM for SDPs with inequality constraints. Moreover, we reported three different methodologies to retrieve a safe bound on their optimal solutions. The extensive computational experiments had a twofold intent: from the one hand, we were able to identify classes of instances where our implementation is competitive with the state-of-the-art solver `SDPNAL+`, from the other hand we investigated the trade-off between the quality of the valid bounds and the computational burden of the three safe-bound methods, showing that in general a good dual bound - even of moderate quality - can be identified despite the convergence criterion is far to be met.

Currently, we are developing a branch-and-bound framework in `python` for solving the stable set problem employing semidefinite relaxations solved by our implementation of the ADMM, equipped with safe bounds procedures. It would be highly interesting to investigate how the valid bounds produced by these procedure may allow to stop the ADMM prematurely when solving the subproblems, yielding a possible speedup of the implicit enumeration.

Bibliography

[1] E. Adams, M. F. Anjos, F. Rendl, and A. Wiegele. A hierarchy of subgraph projection-based semidefinite relaxations for some NP-hard graph optimization problems. *INFOR: Information Systems and Operational Research*, 53(1):40–48, 2015.

[2] E. Balas, S. Ceria, and G. Cornuéjols. A lift-and-project cutting plane algorithm for mixed 0–1 programs. *Mathematical programming*, 58(1):295–324, 1993.

[3] E. Balas, S. Ceria, G. Cornuejols, and G. Pataki. Polyhedral methods for the maximum clique problem. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA GRADUATE SCHOOL OF INDUSTRIAL ADMINISTRATION, 1994.

[4] D. Bertsimas and J. N. Tsitsiklis. *Introduction to linear optimization*, volume 6. Athena Scientific Belmont, MA, 1997.

[5] S. Bianchi, M. Escalante, G. Nasini, and L. Tunçel. Some advances on Lovász-Schrijver $N_+(\cdot)$ relaxations of the fractional stable set polytope. *Electronic Notes in Discrete Mathematics*, 37:189–194, 2011.

[6] S. Bianchi, M. Escalante, G. Nasini, and L. Tunçel. Lovász–schrijver sdp-operator, near-perfect graphs and near-bipartite graphs. *Mathematical Programming*, 162(1):201–223, 2017.

[7] I. M. Bomze, M. Budinich, P. M. Pardalos, and M. Pelillo. The maximum clique problem. In *Handbook of combinatorial optimization*, pages 1–74. Springer, 1999.

[8] S. Burer and R. D. C. Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical Programming*, 95(2, Ser. B):329–357, 2003.

[9] S. Burer and R. D. C. Monteiro. Local minima and convergence in low-rank semidefinite programming. *Mathematical Programming*, 103(3, Ser. A):427–444, 2005.

[10] S. Burer and D. Vandenbussche. Solving lift-and-project relaxations of binary integer programs. *SIAM Journal on Optimization*, 16(3):726–750, 2006.

[11] A. Cabello, S. Severini, and A. Winter. Graph-theoretic approach to quantum correlations. *Physical review letters*, 112(4):040401, 2014.

[12] M. Campêlo, R. Corrêa, and Y. Frota. Cliques, holes and the vertex coloring polytope. *Information Processing Letters*, 89(4):159–164, 2004.

[13] A. Caprara and J. J. S. González. Separating lifted odd-hole inequalities to solve the index selection problem. *Discrete Applied Mathematics*, 92(2-3):111–134, 1999.

[14] M. Cerulli, M. De Santis, E. Gaar, and A. Wiegele. Improving ADMMs for solving doubly nonnegative programs through dual factorization. *4OR*, 19(3):415–448, 2021.

[15] C. Chen, B. He, Y. Ye, and X. Yuan. The direct extension of ADMM for multi-block convex minimization problems is not necessarily convergent. *Mathematical Programming*, 155(1-2):57–79, 2016.

[16] S. Dash. *On the matrix cuts of Lovász and Schrijver and their use in Integer Programming.* Rice University, 2001.

[17] M. De Santis, F. Rendl, and A. Wiegele. Using a factored dual in augmented Lagrangian methods for semidefinite programming. *Operations Research Letters*, 46(5):523 – 528, 2018.

[18] E. Dolan and J.Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91:201–213, 2002.

[19] I. Dukanovic and F. Rendl. Semidefinite programming relaxations for graph coloring and maximal clique problems. *Mathematical Programming*, 109(2):345–365, 2007.

[20] E. Gaar and F. Rendl. A computational study of exact subgraph based SDP bounds for Max-Cut, stable set and coloring. *Mathematical Programming*, 183(1):283–308, 2020.

[21] L. Galli and A. N. Letchford. On the Lovász theta function and some variants. *Discrete Optimization*, 25:159–174, 2017.

[22] M. Giandomenico and A. N. Letchford. Exploring the relationship between max-cut and stable set relaxations. *Mathematical programming*, 106(1):159–175, 2006.

[23] M. Giandomenico, A. N. Letchford, F. Rossi, and S. Smriglio. An application of the Lovász–Schrijver $M(K, K)$ operator to the stable set problem. *Mathematical programming*, 120(2):381–401, 2009.

[24] M. Giandomenico, F. Rossi, and S. Smriglio. Strong lift-and-project cutting planes for the stable set problem. *Mathematical Programming*, 141(1):165–192, 2013.

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

[26] N. Gould and J. Scott. A note on performance profiles for benchmarking software. *ACM Transactions on Mathematical Software (TOMS)*, 43(2):1–5, 2016.

[27] M. Grötschel, L. Lovász, and A. Schrijver. Relaxations of vertex packing. *Journal of Combinatorial Theory, Series B*, 40(3):330–343, 1986.

[28] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric algorithms and combinatorial optimization*, volume 2. Springer Science & Business Media, 2012.

[29] G. Gruber and F. Rendl. Computational experience with stable set relaxations. *SIAM Journal on Optimization*, 13(4):1014–1028, 2003.

[30] S. Gualandi and F. Malucelli. Exact solution of graph coloring problems via constraint programming and column generation. *INFORMS Journal on Computing*, 24(1):81–100, 2012.

[31] L. Gurobi Optimization. Gurobi optimizer reference manual, 2021.

[32] N. Gvozdenović and M. Laurent. Computing Semidefinite Programming Lower Bounds for the (Fractional) Chromatic Number Via Block-Diagonalization. *SIAM Journal on Optimization*, 19(2):592–615, 2008.

[33] N. Gvozdenović and M. Laurent. The Operator Ψ for the Chromatic Number of a Graph. *SIAM Journal on Optimization*, 19(2):572–591, 2008.

[34] J. Hastad. Clique is hard to approximate within $n^{1-\epsilon}$. In *Proceedings of 37th Conference on Foundations of Computer Science*, pages 627–636. IEEE, 1996.

[35] C. Helmberg. Semidefinite programming. *European Journal of Operational Research*, 137(3):461–482, 2002.

[36] A. Jabrayilov and P. Mutzel. New integer linear programming models for the vertex coloring problem. In *Latin american symposium on theoretical informatics*, pages 640–652. Springer, 2018.

[37] A. Jabrayilov and P. Mutzel. Strengthened Partial-Ordering Based ILP Models for the Vertex Coloring Problem. *arXiv preprint arXiv:2206.13678*, 2022.

[38] C. Jansson, D. Chaykin, and C. Keil. Rigorous error bounds for the optimal value in semidefinite programming. *SIAM Journal on Numerical Analysis*, 46(1):180–200, 2008.

[39] D. J. Johnson and M. A. Trick, editors. *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, Workshop, October 11-13, 1993.* American Mathematical Society, 1996.

[40] F. Juhász. The asymptotic behaviour of Lovász'theta function for random graphs. *Combinatorica*, 2(2):153–155, 1982.

[41] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

[42] J. E. Kelley, Jr. The cutting-plane method for solving convex programs. *Journal of the society for Industrial and Applied Mathematics*, 8(4):703–712, 1960.

[43] M. Larsen, J. Propp, and D. Ullman. The fractional chromatic number of Mycielski's graphs. *Journal of Graph Theory*, 19(3):411–416, 1995.

[44] J. B. Lasserre. An explicit exact SDP relaxation for nonlinear 0-1 programs. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 293–303. Springer, 2001.

[45] M. Laurent. A generalization of antiwebs to independence systems and their canonical facets. *Mathematical Programming*, 45(1):97–108, 1989.

[46] M. Laurent. A comparison of the Sherali-Adams, Lovász-Schrijver, and Lasserre relaxations for 0–1 programming. *Mathematics of Operations Research*, 28(3):470–496, 2003.

[47] M. Laurent and F. Rendl. Semidefinite programming and integer programming. In K. Aardal, G. Nemhauser, and R. Weismantel, editors, *Discrete Optimization*, volume 12 of *Handbooks in Operations Research and Management Science*, chapter 8, pages 393–514. Elsevier, Amsterdam, The Netherlands, 2005.

[48] M. Laurent and F. Rendl. Semidefinite programming and integer programming. *Handbooks in Operations Research and Management Science*, 12:393–514, 2005.

[49] A. N. Letchford, F. Rossi, and S. Smriglio. The stable set problem: Clique and nodal inequalities revisited. *Computers & Operations Research*, 123:105024, 2020.

[50] M. Locatelli. Improving upper bounds for the clique number by non-valid inequalities. *Mathematical Programming*, 150(2):511–525, 2015.

[51] D. A. Lorenz and Q. Tran-Dinh. Non-stationary Douglas–Rachford and alternating direction method of multipliers: adaptive step-sizes and convergence. *Computational Optimization and Applications*, 74(1):67–92, Sep 2019.

[52] L. Lovász. On the Shannon capacity of a graph. *IEEE Transactions on Information theory*, 25(1):1–7, 1979.

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

[54] L. Lovász. Kneser's conjecture, chromatic number, and homotopy. *Journal of Combinatorial Theory, Series A*, 25(3):319–324, 1978.

[55] E. Malaguti and P. Toth. A survey on vertex coloring problems. *International transactions in operational research*, 17(1):1–34, 2010.

[56] J. Malick, J. Povh, F. Rendl, and A. Wiegele. Regularization methods for semidefinite programming. *SIAM Journal on Optimization*, 20(1):336–356, 2009.

[57] A. Mehrotra and M. A. Trick. A column generation approach for graph coloring. *informs Journal on Computing*, 8(4):344–354, 1996.

[58] I. Méndez-Díaz and P. Zabala. A Branch-and-Cut algorithm for graph coloring. *Discrete Applied Mathematic*, 154:826–847, 2006.

[59] P. Meurdesoif. Strengthening the Lovász $\theta(\bar{G})$ bound for graph coloring. *Mathematical programming*, 102(3):577–588, 2005.

[60] A. T. Murray and R. L. Church. Facets for node packing. *European Journal of Operational Research*, 101(3):598–608, 1997.

[61] I. Méndez-Díaz and P. Zabala. A cutting plane algorithm for graph coloring. *Discrete Applied Mathematics*, 156(2):159–179, 2008.

[62] Y. Nesterov and A. Nemirovskii. *Interior-point polynomial algorithms in convex programming.* SIAM, 1994.

[63] M. W. Padberg. On the facial structure of set packing polyhedra. *Mathematical programming*, 5(1):199–215, 1973.

[64] J. Povh, F. Rendl, and A. Wiegele. A boundary point method to solve semidefinite programs. *Computing*, 78(3):277–286, 2006.

[65] J. Povh, F. Rendl, and A. Wiegele. A Boundary Point Method to solve Semidefinite Programs. *Computing*, 78:277–286, 2006.

[66] R. T. Rockafellar. *Convex analysis*, volume 18. Princeton university press, 1970.

[67] F. Rossi and S. Smriglio. A branch-and-cut algorithm for the maximum cardinality stable set problem. *Operations Research Letters*, 28(2):63–74, 2001.

[68] F. Rossi and S. Smriglio. A set packing model for the ground holding problem in congested networks. *European Journal of Operational Research*, 131(2):400–416, 2001.

[69] A. Schrijver. A comparison of the Delsarte and Lovász bounds. *IEEE Transactions on Information Theory*, 25(4):425–429, 1979.

[70] J. Schwiddessen. Solving integer quadratic programs using semidefinite optimization, 2023. University of Klagenfurt, Austria. In preparation.

[71] H. D. Sherali and W. P. Adams. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM Journal on Discrete Mathematics*, 3(3):411–430, 1990.

[72] N. J. Sloane. On single-deletion-correcting codes. *Codes and designs*, 10:273–291, 2002.

[73] D. Sun, K.-C. Toh, and L. Yang. A Convergent 3-Block SemiProximal Alternating Direction Method of Multipliers for Conic Programming with 4-Type Constraints. *SIAM Journal on Optimization*, 25:882–915, 2015.

[74] M. Szegedy. A note on the θ Number of lovász and the Generalized Delsarte Bound. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, SFCS '94, page 36–39, USA, 1994. IEEE Computer Society.

[75] L. E. Trotter Jr. A class of facet producing graphs for vertex packing polyhedra. *Discrete Mathematics*, 12(4):373–388, 1975.

[76] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM review*, 38(1):49–95, 1996.

[77] Z. Wen, D. Goldfarb, and W. Yin. Alternating direction augmented Lagrangian methods for semidefinite programming. *Mathematical Programming Computation*, 2(3):203–230, 2010.

[78] A. Wiegele and S. Zhao. Sdp-based bounds for graph partition via extended admm. *arXiv preprint arXiv:2105.09075*, 2021.

[79] A. Wiegele and S. Zhao. Sdp-based bounds for graph partition via extended admm. *Computational Optimization and Applications*, 82(1):251–291, 2022.

[80] L. A. Wolsey and G. L. Nemhauser. *Integer and combinatorial optimization*, volume 55. John Wiley & Sons, 1999.

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

[82] L. Yang, D. Sun, and K.-C. Toh. SDPNAL+: a majorized semismooth Newton-CG augmented Lagrangian method for semidefinite programming with nonnegative constraints. *Mathematical Programming Computation*, 7(3):331–366, 2015.

[83] E. A. Yildirim and X. Fan-Orzechowski. On extracting maximum stable sets in perfect graphs using Lovász's theta function. *Computational Optimization and Applications*, 33(2):229–247, 2006.