# SpeeDP: A new algorithm to compute the SDP relaxations of Max-Cut for very large graphs

L. Grippo L. Palagi M. Piacentini V. Piccialli
G. Rinaldi

September 6, 2010

**Abstract**

We consider low-rank semidefinite programming (LRSDP) relaxations of unconstrained $\{-1, 1\}$ quadratic problems (or, equivalently, of Max-Cut problems) that can be formulated as the nonconvex nonlinear programming problem of minimizing a quadratic function subject to separable quadratic equality constraints. We prove the equivalence of the LRSDP problem with the unconstrained minimization of a new merit function and we define an efficient and globally convergent algorithm, called `SpeeDP`, for finding critical points of the LRSDP problem. We provide evidence of the effectiveness of `SpeeDP` by comparing it with other existing codes on an extended set of instances of the Max-Cut problem.

When the rank of solution matrix is bounded by a given value (independent on the problem size $n$), `SpeeDP` is still able to provide a valid upper bound for Max-Cut. This feature makes it possible to design an algorithm, called `SpeeDP-MC` and based on the Goemans-Williamson heuristic, that has two interesting features: (a) it provides heuristic solutions to Max-Cut along with a guaranteed optimality error; (b) it runs with a $\mathcal{O}(n + m)$ memory requirement (where $m$ is the number of edges of the graph), thus overcoming a serious drawback of interior point based methods that demand $\mathcal{O}(n^2)$ memory. Exploiting the latter feature, we could run it on very large graphs with sizes of up to a million nodes, obtaining very small optimality error bounds in reasonable computation times.

**keywords** Semidefinite programming, low rank factorization, unconstrained binary quadratic programming, Max-Cut, nonlinear programming.

## 1 Introduction

We consider a semidefinite programming (SDP) problem in the form

$$\min_X \{Q \bullet X : \mathrm{diag}(X) = e, X \succeq 0\}, \tag{1}$$

where $Q \in \mathscr{S}^n$ is given, $X \in \mathscr{S}^n$ ($\mathscr{S}^n$ being the space of the $n \times n$ symmetric matrices), $e \in \mathbb{R}^n$ is the vector of all ones, $\mathrm{diag}(X)$ denotes the $n$-vector corresponding to the main diagonal of $X$, and, finally, $X \succeq 0$ tells that $X$ is required to be positive semidefinite.

Semidefinite Programming problems of this form arise as relaxations of unconstrained $\{-1, 1\}$ quadratic problems (see, e.g., [6], [10], [19]):

$$\min_x \left\{ x^T Q x : x \in \{-1, 1\}^n \right\}, \tag{2}$$

which are equivalent to the Max-Cut problem. Given the weighted adjacency matrix $A$ of a weighted graph $G = (V, E)$, the Max-Cut problem calls for a bipartition $(S, V \setminus S)$ of its vertices $V$ so that the weight of the cut, i.e., of the edges joining the two sets of the bipartition, is maximized. Denote by $L$ the Laplacian matrix associated with $A$ and defined by $L := \mathrm{diag}(Ae) - A$. Represent each bipartition $(S, V \setminus S)$ by an $n$-vector defined by $x_i = 1$ for $i \in S$ and $x_i = -1$ for $i \notin S$ (or, equivalently, by its opposite vector). Then the Max-Cut problem can be formulated as

$$\max_x \left\{ \frac{1}{4} x^T L x : x \in \{-1, 1\}^n \right\}. \tag{3}$$

Since $x^T L x = L \bullet x x^T$ and $x x^T \succeq 0$ with $\mathrm{diag}(x x^T) = e$, it is clear that problem (1) with $Q = -\frac{1}{4} L$ provides a relaxation of Max-Cut. Efficient solution of problem (1) is then of great interest because it can be exploited for solving the corresponding integer problem (2) exactly or as a tool for defining good heuristics (see, e.g., [23], [25]).

The aim of this paper is twofold: on one side is to define an efficient algorithm for solving large scale instances of problem (1); on the other, exploiting this useful tool, is to find good solutions of problem (3), by defining a new heuristic algorithm and providing a measure of the distance of the solution weight from the optimal value.

It is well known that problem (1) can be solved by any interior point method, where the key idea consists in applying the Newton method to the optimality conditions of the primal-dual pair of problems.

The dual of problem (1) is

$$\min_y \left\{ e^T y : \mathrm{Diag(y)} + \mathrm{Q} \succeq 0 \right\}. \tag{4}$$

Unfortunately, the interior point methods require $\mathscr{O}(n^2)$ memory which makes it prohibitive to attack instances with, say, $n > 50\,000$. Moreover these methods typically require to perform a Cholesky factorization of a $n \times n$ matrix which require $\mathscr{O}(n^3)$ operations, a much too expensive task when the graph is very large. These limitations motivate searching for methods that are less demanding in terms of memory allocation and avoid the need of the Cholesky factorization. For this reason, the special structure of the constraints of problem (1) has been exploited in the literature to define ad-hoc algorithms. One possibility is to

eliminate the semidefiniteness constraint Diag(y) + Q $\succeq 0$ in the dual problem. At the end, the dual problem is reformulated as eigenvalue optimization problem which can be solved by spectral bundle methods [17]. The other option is to use nonlinear programming reformulations that eliminate the semidefiniteness constraint from the primal problem (1). The latter is the line of research this paper falls into. Indeed, using the Gramian representation, any given matrix $X \succeq 0$ with rank $r$ can be written as $X = VV^T$, where $V$ is a $n \times r$ real matrix. Therefore the positive semidefiniteness constraint can be eliminated, and problem (1) reduces to the Low Rank SDP formulation (LRSDP)

$$\min_V \left\{ Q \bullet VV^T : \mathrm{diag}(VV^T) = e \right\}. \tag{5}$$

For a fixed value of the rank $r$, problem (5) can be written as a Non Linear Programming problem (NLP$_r$)

$$\min_v \left\{ q_r(v) = \sum_{i=1}^{n} \sum_{j=1}^{n} q_{ij} v_i^T v_j : \|v_i\|^2 = 1, i = 1, \ldots, n \right\}, \tag{6}$$

where $v_i$, $i = 1, \ldots, n$, are the columns of the matrix $V^T$ and $v = \mathrm{vec}(V^T) \in \mathbb{R}^{nr}$. Indeed this formulation was first derived by Goemans-Williamson in [12], by replacing each variable $x_i$ of problem (2) with a vector $v_i \in \mathbb{R}^n$ (or $v_i \in \mathbb{R}^r$ with $r \leq n$), obtaining problem (6). Although reformulation (5) results in the non convex problem (6), it is still possible to state conditions that ensure correspondence among global solutions of problem (6) and solutions of problem (1) and also optimality conditions which can be used to check global optimality [14], [21], [15], [5].

In most of the papers based on NLP approaches, the solution of problem (6) is achieved by means of an unconstrained reformulation (see [5], [15], [18] and Section 5).

Indeed, the first idea of an unconstrained formulation of problem (1) goes back to Homer and Peinado [18], but the dimension of the resulting problem made the method prohibitive for large scale problems. Burer and Monteiro in [5] combine the Homer and Peinado formulation with the "low rank idea". By introducing the change of variables $X_{ij} = v_i^T v_j / \|v_i\| \|v_j\|$, where $v_i \in \mathbb{R}^r$, $i = 1, \ldots, n$, with $r << n$, they get the unconstrained formulation

$$\min_v \left\{ f_r(v) = \sum_{i=1}^{n} \sum_{j=1}^{n} q_{ij} \frac{v_i^T v_j}{\|v_i\| \|v_j\|}, \quad v_i \in \mathbb{R}^r \right\}. \tag{7}$$

The resulting algorithm `SDPLR-MC` was computationally efficient, but the underlying convergence theory was not deeply investigated.

In this paper, we start from the unconstrained formulation (7), to get an enhanced different unconstrained formulation, for which we prove complete equivalence with problem (1). The specific feature of this formulation is that we add to the function $f_r(v)$, where $v \in \mathbb{R}^{nr}$ as in [5], a shifted barrier penalty term that

ensures compactness of the level sets of the new merit function. This allows us to use standard unconstrained optimization algorithms. In particular, we define a globally convergent algorithm based on the nonmonotone Barzilai-Borwein gradient method proposed in [16]. The resulting algorithmic scheme `SpeeDP` outperforms the best existing methods for solving problem (1). By fixing $r$ to a value independent of $n$, Algorithm `SpeeDP` requires $\mathcal{O}(n+m)$ memory; moreover, despite the limitation in the rank of the solution matrix, it is still possible to derive a valid lower bound for problem (1). Therefore, it is possible to produce a lower bound (close to the SDP bound) for very large instances of problem (2). In addition, `SpeeDP` provides in output the Gramiam matrix of a solution $X$ to problem (1). This implies that, once `SpeeDP` has produced a solution, the famous Goemans-Williamson algorithm proposed in [12] can be applied, essentially without any additional computational effort, to find a feasible cut that, in case of nonnegative weights, has weight at most 12.1% away from the weight of the optimum. Therefore, we designed a heuristic algorithm exploiting this feature in order to produce good cuts for very large graphs. In this algorithm the Goemans-Williamson cut is improved by applying a 1-opt local search and then by solving problem (1) again a few more times for a perturbed matrix $Q'$. As the computation time is concerned, contrary to what happens for the interior point methods, `SpeeDP` has the ability of exploiting sparsity of matrix $Q$, thus making it possible to find cuts in sparse graphs with millions of nodes and edges, with optimality error lower than 5% (when the edge weights are all positive) in quite practical computation times.

Papers describing heuristics for Max-Cut abound in the literature. However, only in a few cases the algorithms they describe provide a bound on the optimality error for the generated solutions. Excluding the heuristics with a certified a priori bound (like the one of Goemans and Williamson) the only cases when this bound is computed are those of the exact algorithms, that compute an upper bound on the value of optimal solution, by solving a relaxation of the problem. If these algorithms are interrupted prematurely, they provide, as a side product, a heuristic cut along with an upper bound on the optimal value. Unfortunately, the computational studies based on these types of algorithms consider graphs much smaller than those used for the test bed of this paper (see, e.g., [20] for the polyhedral relaxations and [25] for a combination of SDP and polyhedral relaxation), therefore no comparison of the computational results provided here with other approaches is possible at the time.

The paper is structured as follows: in Section 2 we report some useful results about the low rank reformulation of problem (1). In Section 3 we define the new unconstrained reformulation of problem (LRSDP), while in Section 4 we define formally the solution algorithm `SpeeDP` employed for solving this formulation.

In Section 5 we define our heuristic for finding good solutions of large sparse instances of Max-Cut.

In Section 6 we report the numerical results. We compare the performance of `SpeeDP` against other existing approaches for problem (1). Then we use the heuristic to find good solutions of large and huge instances of the Max-Cut problem for random graphs.

Throughout the paper, given an $m \times p$ matrix $M$ we denote by $\mathrm{vec}(M) \in \mathbb{R}^{mp}$ the vector corresponding to the elements of $M$ ordered by column index and then by row index. Given a vector $v \in \mathbb{R}^m$, we denote by $\mathrm{Diag}(v)$ the diagonal matrix having as diagonal the vector $v$ and by $B_\rho(v)$ the closed ball centered in $v$ with radius $\rho > 0$, namely $B_\rho(v) = \{y \in \mathbb{R}^m : \|y - v\| \leq \rho\}$. For a given scalar $x$ we denote by $(x)_+$ the maximum between $x$ and zero, namely $(x)_+ \equiv \max(x, 0)$.

# 2 Some useful results about the low rank SDP formulation

In this section we report the main results on the Low Rank SDP formulation (LRSDP) defined in (5).

A global minimum point of problem (5) is a solution of problem (1) provided that

$$r \geq r_{\min} = \min_{X \in \mathscr{X}^*_{\mathrm{SDP}}} \mathrm{rank}(X),$$

where $\mathscr{X}^*_{\mathrm{SDP}}$ denotes the optimal solution set of problem (1). Although the value of $r_{\min}$ is not known, an upper bound can easily be computed by exploiting the result proved in [24], [2], [22], that gives

$$r_{\min} \leq \widehat{r} = \frac{\sqrt{8n+1} - 1}{2}. \tag{8}$$

Thus, in order to get a problem equivalent to problem (1), the dimension of the matrix $V$ in problem (5) can be fixed to $n \times r$ with $r \geq \widehat{r}$.

We say that *a point* $v^* \in \mathbb{R}^{nr}$ *solves problem* (1) if $X^* = V^* V^{*T}$ is an optimal solution of problem (1). This implies, by definition, that $r \geq r_{\min}$.

Although reformulation (5) results in the non convex problem (6), the primal-dual optimality conditions for (1) combined with necessary optimality conditions for (6) lead to some global optimality conditions that can be exploited from the computational point of view [14], [21], [15], [5].

The Karush-Kuhn-Tucker conditions for problem (6) are written as follows for some $\lambda \in \mathbb{R}^n$

$$\begin{aligned} &\sum_{j=1}^{n} q_{ij} v_j + \lambda_i v_i = 0, \qquad && i = 1, \ldots, n \\ &\|v_i\|^2 = 1, && i = 1, \ldots, n. \end{aligned} \tag{9}$$

We define stationary point of problem (6) a point $\hat{v} \in \mathbb{R}^{nr}$ satisfying (9) with a suitable multiplier $\hat{\lambda} \in \mathbb{R}^n$. Given a local minimizer $\hat{v} \in \mathbb{R}^{nr}$ of problem (6), the KKT conditions are necessary conditions for optimality and there exists a unique $\hat{\lambda} \in \mathbb{R}^n$ such that $(\hat{v}, \hat{\lambda})$ satisfies (9). This feature has been exploited first in [15] and later for slightly more general constraints in [21]. Indeed, given

a pair $(v, \lambda)$ satisfying the conditions (9), the multiplier $\lambda$ can be expressed uniquely as a function of $v$, namely

$$\lambda_i(V) = \lambda_i = -E_{ii}Q \bullet VV^T = -{v_i}^T \sum_{j=1}^{n} q_{ij} v_j, \quad i = 1, \ldots, n. \tag{10}$$

By substituting the expression of $\lambda$ in the first condition of (9), we get

$$\sum_{j=1}^{n} q_{ij} \left(I_r - v_i v_i^T\right) v_j = 0 \quad i = 1, \ldots, n. \tag{11}$$

The next proposition that extends the sufficient conditions given in [5] and was proved in in [15] and for more general problems in [14], [21], states the global optimality conditions obtained by exploiting the primal-dual properties for problem (1).

**Proposition 3** (Global optimality conditions)**.** *A point $v^* \in \mathbb{R}^{nr}$ is a global minimizer of problem* (6) *that solves problem* (1) *if and only if it is a stationary point of problem* (6) *and satisfies*

$$Q + \mathrm{Diag}(\lambda(\mathrm{V}^*)) \succeq 0,$$

*where $\lambda(V^*)$ is computed according to* (10)*.*

Thanks to the above proposition, given a stationary point of problem (6), we can prove its optimality just checking that a certain matrix is positive semidefinite.

Another global condition has been proved in [21] for a slightly more general convex SDP problem which includes as a special case problem (1). It is proved that the $n \times r$ local minimizer $\widehat{V}$ of the LRSDP problem provides a global solution $\widehat{X} = \widehat{V}\widehat{V}^T$ of the original SDP problem if $\widehat{V}$ is rank deficient, namely if $rank(\widehat{V}) < r$. Actually looking at the proof, it turns out that the assumption of $\widehat{V}$ being a local minimizer can be relaxed to satisfying the second order necessary conditions for the LRSDP problem. Hence we restate their proposition (only in the special case of problem (1), although a generalization to linear constraints and convex objective function easily follows).

**Proposition 4.** *Let $\widehat{V}$ be the $n \times r$ matrix satisfying the first and second order necessary conditions of problem* (5)*, namely* (9) *and*

$$\left(Q + \mathrm{Diag}(\widehat{\lambda})\right) \bullet ZZ^T \geq 0 \qquad \textit{for all } Z \in \mathbb{R}^{n \times r} : \quad E_{ii} \bullet \widehat{V}Z^T = 0 \quad i = 1, \ldots, n.$$

*If the matrix $\widehat{V}$ is rank deficient, then it provides a global solution $\widehat{X} = \widehat{V}\widehat{V}^T$ of problem* (1)*. If $r = n$, any $n \times n$ matrix $\widehat{V}$ satisfying the second order necessary conditions of problem* (5) *provides a global solution $\widehat{X} = \widehat{V}\widehat{V}^T$ of problem* (1)*.*

For sake of completeness we report the proof in Appendix 14 although it is only a special case of the proof presented in [21]. We note for $r = n$ it was already proved in [18] that there exists no local minimizer $\widehat{V}$ of problem (5) which is not global.

# 5 A new unconstrained formulation of the SDP problem

In most papers considering problem (6), its solution is achieved by means of an unconstrained reformulation of it. In particular, the augmented Lagrangian function proposed in [5] for a semidefinite programming problem with general linear constraints can be specified to problem (1) and takes the form

$$\mathscr{L}(V, \lambda; \varepsilon) = Q \bullet VV^T + \sum_{i=1}^{n} \lambda_i \left( \|v_i\|^2 - 1 \right) + \frac{1}{2\varepsilon} \sum_{i=1}^{n} \left( \|v_i\|^2 - 1 \right)^2,$$

where $\varepsilon > 0$ is a penalty parameter and $\lambda_i \in \mathbb{R}$, $i = 1, \ldots, n$. This function is minimized for a sequence of suitable values of $(\lambda^k, \varepsilon^k)$, where $\varepsilon^k$ is increasing and $\lambda^k$ is obtained with some updating rule. In [15], the structure of the constraints in problem (6) has been exploited to get the closed expression (10) of the multipliers $\lambda_i(V)$ as a function of the variables $v$. Replacing the multipliers $\lambda_i$ with the closed expression $\lambda_i(V)$ in the augmented Lagrangian function $\mathscr{L}(V, \lambda; \varepsilon)$ and fixing the penalty parameter to a value $\overline{\varepsilon} > 0$, they get an exact penalty function

$$P(V) = Q \bullet VV^T + \sum_{i=1}^{n} \lambda_i(V) \left( \|v_i\|^2 - 1 \right) + \frac{1}{2\overline{\varepsilon}} \sum_{i=1}^{n} \left( \|v_i\|^2 - 1 \right)^2.$$

In this case a single unconstrained minimization of the twice continuously differentiable function $P(V)$ is enough to find a stationary point of problem (6). Computational experiments with the resulting algorithmic scheme, called `EXPA` in [15], showed that this unconstrained approach compares favorably with the best codes available in literature. More recently, Journée et al. in [21] use a trust region method for optimizing over a manifold [1], which relies on a particular quotient manifold. Their algorithm is defined for a slightly more general problem than (1) since they consider a generic convex objective function and general linear constraints. Their method relies on exploiting the special structure of the constraints to find a closed expression of the multipliers, which in the special case of problem (6) returns the same expression found in [15].

However, the original idea of an unconstrained formulation of problem (1) goes back to Homer and Peinado [18], where the change of variables $X_{ij} = v_i^T v_j / \|v_i\| \|v_j\|$ for the elements of $X$ with $v_i \in \mathbb{R}^n$, $i = 1, \ldots, n$ has been used to formulate an unconstrained optimization problem equivalent to the original problem (6). Their approach led to a problem of dimension $n^2$ which was solved by a parallel gradient method, but turned out to be impractical for large values of $n$.

In [5] the unconstrained formulation proposed by Homer and Peinado has been resumed and combined it with the "low rank idea", by introducing the change of variables $X_{ij} = v_i^T v_j / \|v_i\| \|v_j\|$ where $v_i \in \mathbb{R}^r$, $i = 1, \ldots, n$, with $r < n$. The resulting unconstrained problem (7) was solved to obtain a solution of problem (5). The practical performance of the resulting algorithm, that

we called, SDPLR-MC was pretty good, but the underlying convergence theory was not deeply investigated. Indeed, it is easy to show that problem (7) is equivalent to problem (5) in the sense that a one-to-one correspondence among local/global/stationary point of the two problems can be stated (see [13]). However, problem (7) presents some peculiarities that make standard convergence results not immediately applicable. Indeed, standard unconstrained algorithms can be proved to be globally convergent if the objective function is continuously differentiable and has compact level sets. Function $f_r(v)$ is not even defined at points where $\|v_i\| = 0$ for at least one index $i$. In principle, it is possible to modify standard algorithms by looking not at the sequence $\{(v_1, \ldots, v_n)^k\}$ but at the normalized sequence $\{(v_1/\|v_1\|, \ldots, v_n/\|v_n\|)^k\}$. However, this may cause difficulties in proving convergence of standard optimization algorithms.

In this paper we propose to modify $f_r$ in such a way to get an unconstrained problem that can be solved by standard methods. In particular, we add a shifted barrier penalty term

$$\sum_{i=1}^{n} \frac{(\|v_i\|^2 - 1)^2}{d(v_i)}, \tag{12}$$

where

$$d(v_i) \equiv \delta^2 - \left(1 - \|v_i\|^2\right)_+^2, \quad 0 < \delta < 1. \tag{13}$$

For a fixed $\varepsilon > 0$, we consider the unconstrained minimization problem

$$\min_v \left\{ f_\varepsilon(v) = f_r(v) + \frac{1}{\varepsilon} \sum_{i=1}^{n} \frac{(\|v_i\|^2 - 1)^2}{d(v_i)},\ v \in S_\delta \right\}, \tag{14}$$

where $f_r(v)$ is given in (7) and the open set $S_\delta$ is defined as

$$S_\delta \equiv \{v \in \mathbb{R}^{nr} : \|v_i\|^2 > 1 - \delta, \quad i = 1, \ldots, n\}.$$

The added term (12) ensures that the level sets of $f_\varepsilon$ are contained in the set $S_\delta$ and are compact. Hence, problem (14) allows us to overcome all the theoretical drawbacks of problem (7). In particular, we will show that solving problem (14) for a single value of $\varepsilon$ is equivalent to solving problem (6).

We start by investigating the theoretical properties of the function $f_\varepsilon(v)$. Function $f_\varepsilon(v)$ is continuously differentiable on the open set $S_\delta$ with gradient

$$\nabla_{v_i} f_\varepsilon(v) = \nabla_{v_i} f_r(v) + \frac{4}{\varepsilon} \frac{(\|v_i\|^2 - 1)}{d(v_i)} \left[1 - \frac{(\|v_i\|^2 - 1)(1 - \|v_i\|^2)_+}{d(v_i)}\right] v_i$$

where

$$\nabla_{v_i} f_r(v) = \frac{2}{\|v_i\|} \left[\sum_{j=1}^{n} q_{ij} \left(I_r - \frac{v_i}{\|v_i\|} \frac{{v_i}^T}{\|v_i\|}\right) \frac{v_j}{\|v_j\|}\right].$$

The first important property is the compactness of the level sets of function $f_\varepsilon(v)$, that guarantees the existence of a solution of problem (14).

**Proposition 6.** *For every given $\varepsilon > 0$ and for every given $v^0 \in S_\delta$, the level sets $\mathscr{L}_\varepsilon(v^0) = \{v \in S_\delta : \ f_\varepsilon(v) \leq f_\varepsilon(v^0)\}$ are compact and*

$$\mathscr{L}_\varepsilon(v^0) \subseteq \left\{v \in \mathbb{R}^{nr} : \|v_i\|^2 \leq C(\varepsilon\delta), \quad i = 1, \ldots, n\right\},$$

*with $C(\varepsilon\delta) > 0$ positive constant depending from $\varepsilon$ and $\delta$.*

The proof can be found in Appendix 14.

An interesting property of the objective function $f_r(v)$ of problem (7) is that, given a point $v$ in $S_\delta$, its gradient with respect to $v_i$ is orthogonal to the vector $v_i$, namely, for every $v \in S_\delta$ and for every $i = 1, \ldots, n$

$$v_i^T \nabla_{v_i} f_r(v) = 2\left[\sum_{j=1}^{n} q_{ij}\left(\frac{v_i^T}{\|v_i\|} - \frac{v_i^T v_i}{\|v_i\|^2}\frac{{v_i}^T}{\|v_i\|}\right)\frac{v_j}{\|v_j\|}\right] = 0. \tag{15}$$

The following theorem states the equivalence between stationary points, local/global minimizers of (14) and the corresponding stationary points, local/global minimizers of (6).

**Theorem 7** (Exactness properties of (14))**.** *For any $\varepsilon > 0$ and for any fixed $r \geq 1$, the following correspondences hold:*

(*i*) *a point $\hat{v} \in \mathbb{R}^{nr}$ is a stationary point of problem* (14) *if and only if it is a stationary point of problem* (6)*.*

(*ii*) *a point $\hat{v} \in \mathbb{R}^{nr}$ is a global minimizer of problem* (14) *if and only if it is a global minimizer of problem* (6)*.*

(*iii*) *a point $\hat{v} \in \mathbb{R}^{nr}$ is a local minimizer of problem* (14) *if and only if it is a local minimizer of problem* (6)*.*

*Proof.* First, we recall that, for every $v \in S_\delta$, $v_i \neq 0$ for all $i = 1, \ldots, n$. Furthermore, by definition of $\nabla_{v_i} f_\varepsilon$ and by (15), we get for every $v_i$ and for $i = 1, \ldots, n$

$$v_i^T \nabla_{v_i} f_\varepsilon(v) = \frac{4}{\varepsilon}\frac{(\|v_i\|^2 - 1)v_i^T v_i}{d(v_i)}\left(1 - \frac{(\|v_i\|^2 - 1)(1 - \|v_i\|^2)_+}{d(v_i)}\right).$$

Therefore we get, if $\|v_i\|^2 \geq 1$,

$$v_i^T \nabla_{v_i} f_\varepsilon(v) = \frac{4}{\varepsilon}\frac{(\|v_i\|^2 - 1)\|v_i\|^2}{\delta^2}, \tag{16}$$

otherwise

$$v_i^T \nabla_{v_i} f_\varepsilon(v) = \frac{4}{\varepsilon}\frac{(\|v_i\|^2 - 1)\|v_i\|^2}{d(v_i)}\left(1 + \frac{(\|v_i\|^2 - 1)^2}{d(v_i)}\right). \tag{17}$$

Furthermore, if $v \in \mathscr{F}$

$$f_\varepsilon(v) = f_r(v) = q_r(v) \tag{18}$$

$$\nabla_{v_i} f_\varepsilon(v) = 2\sum_{j=1}^{n} q_{ij}(I_r - v_i {v_i}^T)v_j, \quad i = 1, \ldots, n. \tag{19}$$

Now we can prove the three statements.

(*i*) *Sufficiency.* Let $\hat{v}$ be a stationary point for problem (6). Therefore $\hat{v}$ satisfies (11) and $\hat{v} \in \mathscr{F}$. Then (19) implies

$$\nabla_{v_i} f_\varepsilon(\hat{v}) = 2\sum_{j=1}^{n} q_{ij}(I_r - \hat{v}_i \hat{v}_i^T)\hat{v}_j = 0, \quad i = 1, \ldots, n.$$

(*i*) *Necessity.* By (16) and (17), $\hat{v} \in S_\delta$ being a stationary point of $f_\varepsilon$ implies $\hat{v} \in \mathscr{F}$. Hence, as a result of (19), $\hat{v}$ is stationary point also for problem (6).

(*ii*) *Necessity.* By Proposition 6, the function $f_\varepsilon$ admits a global minimizer $\hat{v}$, which is obviously a stationary point of $f_\varepsilon$ and hence we have that $\hat{v} \in \mathscr{F}$, so that $f_\varepsilon(\hat{v}) = q_r(\hat{v})$. We proceed by contradiction. Assume that a global minimizer $\hat{v}$ of $f_\varepsilon$ is not a global minimizer of problem (6). Then there exists a point $v^* \in \mathscr{F}$, global minimizer of problem (6), such that

$$f_\varepsilon(\hat{v}) = q_r(\hat{v}) > q_r(v^*) = f_\varepsilon(v^*),$$

but this contradicts the assumption that $\hat{v}$ is a global minimizer of $f_\varepsilon$.

(*ii*) *Sufficiency.* True by similar arguments.

(*iii*) *Necessity.* Since $\hat{v}$ is a local minimizer of $f_\varepsilon$, it is a stationary point of $f_\varepsilon$, so that $\hat{v} \in \mathscr{F}$. Thus, $f_\varepsilon(\hat{v}) = q_r(\hat{v})$. Since $\hat{v}$ is a local minimizer of $f_\varepsilon$, there exists a $\rho > 0$ such that for all $v \in S_\delta \cap B_\rho(\hat{v})$ such that

$$q_r(\hat{v}) = f_\varepsilon(\hat{v}) \leq f_\varepsilon(v).$$

Therefore, by using (18), for all $v \in v \in \mathscr{F} \cap B_\rho(\hat{v})$ we have that

$$q_r(\hat{v}) \leq f_\varepsilon(v) = q_r(v).$$

and hence $\hat{v}$ is a local minimizer for problem (6).

(*iii*) *Sufficiency.* Since $\hat{v} \in \mathscr{F}$ and is a local minimizer of (6), there exists a $\rho > 0$ such that for all $v \in \mathscr{F} \cap B_\rho(\hat{v})$

$$q_r(\hat{v}) = f_\varepsilon(\hat{v}) \leq q_r(v) = f_\varepsilon(v).$$

We want to show that there exists $\gamma$ such that for all $v \in S_\delta \cap B_\gamma(\hat{v})$ we get

$$f_\varepsilon(\hat{v}) \leq f_\varepsilon(v).$$

It is sufficient to show that there is a $\gamma > 0$ such that for all $v \in S_\delta \cap B_\gamma(\hat{v})$, we have that $p(v) \in B_\gamma(\hat{v})$, where

$$p(v) \equiv \begin{pmatrix} \frac{v_1}{\|v_1\|} \\ \vdots \\ \frac{v_n}{\|v_n\|} \end{pmatrix}.$$

Actually in this case we have

$$q_r(\hat{v}) = f_\varepsilon(\hat{v}) \leq q_r(p(v)) = f_\varepsilon(p(v)) \leq f_\varepsilon(v).$$

It is well known that, given any point $x \neq 0 \in \mathbb{R}^n$, its projection over the unit norm set is simply $\frac{x}{\|x\|}$. Hence, for any $\gamma \leq \frac{\rho}{2}$ we can write

$$\begin{aligned} \|p(v) - \hat{v}\|^2 &= \sum_{i=1}^n \|\hat{v}_i - \frac{v_i}{\|v_i\|}\|^2 = \sum_{i=1}^n \|\hat{v}_i - \frac{v_i}{\|v_i\|} + v_i - v_i\|^2 \\ &\leq \sum_{i=1}^n \left( \|\hat{v}_i - v_i\|^2 + \|v_i - \frac{v_i}{\|v_i\|}\|^2 + 2\|\hat{v}_i - v_i\|\|v_i - \frac{v_i}{\|v_i\|}\| \right) \\ &\leq \sum_{i=1}^n 4\|\hat{v}_i - v_i\|^2 = 4\|\hat{v} - v\|^2 \leq 4\gamma^2 \leq \rho^2 \end{aligned}$$

Therefore, for a proper $\gamma$, we have for all $v \in S_\delta \cap B_\gamma(\hat{v})$

$$f_\varepsilon(\hat{v}) \leq f_\varepsilon(v),$$

so that $\hat{v}$ is a local minimum also for (14). □

Theorem 7 states a tight relation between problem (14) and (6) and hence allows us to solve problem (6) by minimizing $f_\varepsilon(v)$. We stress that all the properties of problem (14) hold for any given $\varepsilon > 0$.

Proposition 7 implies that we can solve problem (6) by solving problem (14), and Proposition 6 implies that any standard global convergent unconstrained minimization method can be used for solving it (see, e.g., [4] for a complete review of unconstrained algorithms). Since $f_\varepsilon(v)$ is continuously differentiable over the set $S_\delta$ and, by Proposition 6, it has compact level sets, by applying a convergent unconstrained procedure, we can easily state the following convergence result.

**Proposition 8.** *Let $r$ be given and $v^0 \in \mathscr{F}$. Assume we apply to problem* (14) *any unconstrained procedure that produces a sequence $\{v^k\}$ such that (i) $v^k$ stays in the initial level set, (ii) it admits at least an accumulation point, (iii) every accumulation point is a stationary point of the objective function. Then*

(*i*) $\{v^k\}$ *is bounded and it admits at least an accumulation point;*

(*ii*) *every accumulation point is a stationary point of problem* (6)*;*

(*iii*) *if* $\hat{v}$ *is an accumulation point then* $q_r(\hat{v}) \leq q_r(v^0)$.

*Proof.* Function $f_\varepsilon(v)$ is continuously differentiable over the set $S_\delta$ and Proposition 6 implies that it has compact level sets. Therefore the assumptions made on the unconstrained procedure imply that it produces a sequence that has at least an accumulation point and all the accumulation points are stationary points of problem (14). Finally, Theorem 7 implies that the stationary points of $f_\varepsilon$ are stationary points of problem (6), and we have

$$q_r(\hat{v}) = f_\varepsilon(\hat{v}) \leq f_\varepsilon(v^0) = q_r(v^0).$$

□

# 9 SpeeDP: an efficient algorithm for solving the SDP problem

In this section, we define an algorithm for solving problem (1) that exploits the results stated in the previous sections.

In Section 2 we have seen that for $r \geq r_{\min}$ a global solution of problem (6) provides a solution of problem (1). Moreover, Proposition 7 states a complete correspondence between problems (6) and (14). Finally, Proposition 8 ensures that we can find a stationary point of problem (14) by applying any global convergent unconstrained minimization procedure.

In our algorithm we select a nonmonotone gradient method defined by an iteration of the form

$$v_i^{k+1} = v_i^k - \alpha^k \nabla_{v_i} f_\varepsilon(v^k) \quad i = 1, \ldots, n, \tag{20}$$

where $\alpha^k > 0$ is obtained by a suitable line-search procedure satisfying

$$f_\varepsilon(v^{k+1}) \leq f_\varepsilon(v^0), \tag{21}$$

with $v^0 \in \mathscr{F}$.

This choice is motivated by the fact that, using a gradient method and for $\varepsilon$ sufficiently large, the produced sequence stays in the set $\{v \in \mathbb{R}^{nr} : \|v_i\|^2 \geq 1, i = 1, \ldots, n\}$. This result implies that the barrier term (12), that may affect negatively the performance behavior of any optimization method when the produced sequence gets closer to the boundary of $S_\delta$, reduces simply to a penalty term on the feasibility of problem (6).

In particular, the following proposition holds whose proof can be found in Appendix 14.

**Proposition 10.** *Let $v^0 \in \mathscr{F}$ and let $\{v^k\}$ be the sequence generated with the iterative scheme* (20)*, where each $\alpha^k$ satisfies* (21) *and $\alpha^k \leq \alpha_M$. Then, there exists $\bar{\varepsilon} > 0$ such that, for any $\varepsilon \geq \bar{\varepsilon}$, we have for $k = 1, 2, \ldots$*

$$\|v_i^k\| \geq 1, \quad i = 1, \ldots, n.$$

The value of $r_{\min}$ is not known. In principle, the only value of $r$ that can be calculated and that guarantees the correspondence between solutions of (1) and global solutions of (6), is $\widehat{r}$ as defined in (8). However, this value is usually larger than the actual value needed to obtain a solution of problem (1). Hence, following the idea in [5] and [15], we choose $r << \widehat{r}$, and use the global optimality condition of Proposition 3 to prove optimality. We use an incremental rank scheme as in algorithm EXPA defined in [15].

**ALGORITHM** `SpeeDP`

**Initialization.** Set integers $2 \leq r^1 < r^2 < \ldots < r^p$ with $r^p \in [\widehat{r}, n]$ where $\widehat{r}$ is given by (8). Choose $\bar{\varepsilon} > 0$, $\bar{\delta} \in [0, 1]$ and $tol_\varepsilon > 0$.

**For** $j = 1, \ldots, p$ **do:**

S.0 Set $r = r^j$ in problem (5).

S.1 Starting from $V^0 \in \mathbb{R}^{n \times r^j}$, find a stationary point $\widehat{V} \in \mathbb{R}^{n \times r^j}$ of problem (14) with $\varepsilon = \bar{\varepsilon}$ and $\delta = \bar{\delta}$.

S.2 Compute $\lambda(\widehat{V})$ with (10) and the minimum eigenvalue $\lambda_{\min}(\widehat{V})$ of $Q + \mathrm{Diag}(\lambda(\widehat{\mathrm{V}}))$.

S.3 If $\lambda_{\min}(\widehat{V}) \geq -tol_\varepsilon$, then exit.

**Return** $\widehat{V} \in \mathbb{R}^{n \times r^j}$ and $\lambda_{\min}(\hat{V})$

`SpeeDP` returns $\widehat{V}$, and $\lambda_{\min}(\widehat{V})$. If $\lambda_{\min}(\widehat{V}) \geq -tol_\varepsilon$, then the matrix $Q + \mathrm{Diag}(\lambda(\widehat{\mathrm{V}}))$ is positive semidefinite within a tolerance $tol_\varepsilon$ so that a solution of problem (1) is obtained as $X^* = \widehat{V}\widehat{V}^T$. If the optimality condition is not met, namely $\lambda_{\min}(\widehat{V}) < -tol_\varepsilon$, a bound can be easily computed. Indeed, since $\left(\lambda(\widehat{V}) + \lambda_{\min}(\widehat{V})\right) e$ is feasible for the dual problem (4), the value $z_{LB} = Q \bullet \widehat{V}\widehat{V}^T + n\lambda_{\min}(\widehat{V})$, provides a lower bound on the solution of problem (1) (see, e.g, [27], [15]).

In practice, however, in all the computational experiments performed the stopping condition $\lambda_{\min}(\widehat{V}) \geq -tol_\varepsilon$ was always met with satisfactory accuracy, so that `SpeeDP` always converged to a solution of (1), as we will illustrate in the Section 12.

# 11 SpeeDP-MC: a heuristic for large scale Max-Cut

Our heuristic is essentially the one due to Goemans and Williamson and described in [12], integrated with `SpeeDP` and a few simple details.

The Goemans-Williamson algorithm is very well known and actually contributed to making SDP techniques popular; nevertheless, we briefly outline it here for the sake of completeness.

Let $X$ be the optimal solution to (1), $-z_P$ be its optimal value (for the Max-Cut problem the objective function has to be maximized), and let $v_1$, $v_2$, ..., $v_n \in \mathbb{R}^r$ be vectors whose Gramian matrix coincides with $X$. Let $h^T x = 0$ define a hyperplane of $\mathbb{R}^r$ generated by drawing the value of the $r$ components of $h$ from a uniform random distribution. Then the algorithm outputs the node bipartition $(S, V \setminus S)$ where $S = \{i : h^T v_i \geq 0\}$.

We assume here that all components of the weighted adjacency matrix $A$ of $G$ are non-negative. The expected weight $\overline{W}$ of the cut defined by the bipartition $(S, V \setminus S)$ is given by

$$\overline{W} = \sum_{i \in S,\, j \notin S} A_{ij} p_{ij},$$

where $p_{ij}$ is the probability that edge $ij$ belongs to the cut or, equivalently, the probability that $v_i$ and $v_j$ lay on opposite sides with respect to the hyperplane defined by $h^T x = 0$. Such a probability is proportional to the angle defined by the two vectors. Finally, using the inequality $\arccos(\alpha)/\pi \geq 0.87856(1-\alpha)/2$, we can write

$$\overline{W} = \sum_{ij} A_{ij} \frac{\arccos(v_i^T v_j)}{\pi} \geq 0.87856 \sum_{ij} A_{ij} \frac{1 - v_i^T v_j}{2} = 0.87856 z_P.$$

In conclusion the gap (in percentage with respect to the SDP bound) obtained by using the relaxation (1) and the Goemans-Williamson algorithm is around 12.1%.

Once the SDP bound has been computed, the main computational effort of the Goemans-Williamson algorithm, is essentially devoted to finding the vectors $v_i$, with $i = 1, \ldots, n$. This task can be accomplished by a truncated Cholesky decomposition of the matrix $X$ which requires time proportional to $n^3$ and space proportional to $n^2$. Therefore the algorithm cannot be applied to very large instances with size, say, of the order of one hundred thousand nodes.

To the contrary, `SpeeDP` makes it possible to apply the Goemans-Williamson approximation algorithm to very large graphs since on the one hand it is able to solve problem (1) in reasonable time also for very large graphs, and on the other hand, it outputs the vectors $v_i$, avoiding the need of a Cholesky factorization. In our procedure the cut provided by the Goemans-Williamson algorithm is then improved by means of a 1-opt local search, where all possible moves of a single vertex to the opposite set of the partition are checked and moved are made until no further improvement is possible.

In [8], where a similar heuristic is described but problem (1) is solved by interior point algorithm, a particularly successful step is proposed to further improve on the solution. The whole procedure is repeated a few times where the solution matrix $X$ of problem (1) is replaced by the convex combination $X' = \alpha X + (1-\alpha)\hat{x}\hat{x}^T$, $0 < \alpha < 1$, where $\hat{x}$ is the representative vector of the current best cut. The idea behind this step is to bias the Goemans-Williams

rounding with the current best cut, or put it differently, to force the rounding procedure to generate a cut in a neighborhood of the current best solution.

This step does not require to solve problem (1) again, but needs the Cholesky factorization of the matrix $X'$.

We use a similar technique in our procedure. However, to avoid the Cholesky factorization, which is not suitable for very large instances, we solve a new problem (1) after perturbing the objective function. Matrix $Q$ is replaced by the perturbed matrix $Q'$ given by $Q' = Q + \beta\hat{x}\hat{x}^T$ with $\beta > 0$.

Such a perturbation has again the effect of moving the solution of problem (1) and hence of the Goemans-Williamson rounding, towards a neighborhood of the current best integral solution. With the new objective function $Q'$ we solve problem (1) with `SpeeDP` and repeat the rounding and the 1-opt improvement as well. The whole procedure is repeated a few times with different values of $\beta$.

Summarizing, the scheme of our heuristic algorithm is as follows:

**ALGORITHM** `SpeeDP-MC`

**Data:** $Q,\ \hat{x} = e,\ \alpha > 0,\ k_{\max},\ \overline{Q} = \sum_{i,j} |Q_{ij}|/|E|$.

**For** $k = k_{max}, \ldots, 0$ **do** :

S.0 Set $\beta = k\alpha\overline{Q} \cdot J$ and $Q' = Q + \beta(\hat{x}\hat{x}^T)$

S.1 Apply `SpeeDP` to problem (1) with $Q = Q'$ and let $v_i,\ i = 1, \ldots, n$ be the returned solution and the valid bound $\phi$ on problem (3) with objective function corresponding to $Q'$.

S.2 Apply the Goemans-Williamson hyperplane rounding technique to the vectors $v_i,\ i = 1, \ldots, n$. This gives a bipartition representative vector $\bar{x}$.

S.3 Apply the 1-opt improvement to $\bar{x}$. This gives a new bipartition representative vector $\tilde{x}$. If $Q' \bullet \tilde{x}\tilde{x}^T < Q' \bullet \hat{x}\hat{x}^T$, set $\hat{x} = \tilde{x}$.

**Return** Best cut $\hat{x}$, lower bound $-Q' \bullet \hat{x}\hat{x}^T$, upper bound $\phi$.

Note that the amount of perturbation decreases when the iteration counter increases, getting to zero in the last iteration. We stress that Step 1 is not expensive since we use a warm start technique: at each iteration we start `SpeeDP` from the solution found at the previous step, so that each minimization is computationally cheaper than the first one.

Besides the ability of treating graphs of very large sizes, another advantage of `SpeeDP-MC` is that it also provides a solution with a guaranteed optimality error bound, since it outputs an upper and lower bound on the value of the optimal cut.

Numerical results for this heuristic are reported in next section.

# 12 Numerical Results

In this section, we describe our computational experience both with algorithm `SpeeDP` for solving problem (1), and with the heuristic based on it for finding good cuts for large graphs.

`SpeeDP` is implemented in Fortran 90 and all the experiments have been run on a PC with processor Core2 DUO E6750 2.66Ghz, and RAM of 2.00 GB.

The parameter $\delta$ that appears in the definition of the open set $S$ has been set to 0.25. This value has been chosen after some experiments for different values of $\delta$. The parameter $\varepsilon$ is set equal to $10^3/\delta$ for all the tests.

As unconstrained optimization procedure we use a Fortran 90 implementation of the non monotone Barzilai-Borwein gradient method proposed in [16] which falls in the iterative scheme (20), (21) and satisfies the assumption of Proposition 8. The termination criteria in the minimization procedure are standard ones with tolerance in the range $10^{-5}$.

As for the choice of the starting value $r^1$ of the rank we use the same values as in [15], reported in Table 1. The values of the rank $r^j$ are chosen with the simple rule $r^{j+1} = \min\left\{\lfloor r^j \cdot 1.5 \rfloor, \widehat{r}\right\}$ where $\widehat{r}$ is given in (8).

| $n \leq 200$ | $200 < n$ $n \leq 800$ | $800 < n$ $n \leq 1000$ | $1000 < n$ $n \leq 5000$ | $5000 < n$ $n \leq 20000$ | $n > 20000$ |
|---|---|---|---|---|---|
| 8 | 10 | 15 | 18 | 25 | 30 |

Table 1: Values of $r^1$ depending on the dimension of the problem.

In order to check positive semidefiniteness of $Q + \mathrm{Diag}(\lambda(\widehat{\mathrm{V}}))$, we use the ARPACK subroutines `dsaupd` and `dseupd` to compute the minimum eigenvalue of this matrix. We set the tolerance $tol_\varepsilon = -10^{-3}$.

As a first step, we consider `SpeeDP` for solving problem (1). We compare the performance of `SpeeDP` with the best codes in literature in the main classes of methods for solving problem (1): interior point methods, Spectral Bundle methods and low rank NLP methods.

As an interior point method we select the dual-scaling algorithm defined in [3] and implemented in the software `DSDP`(version 5.8) downloaded from the web page `http://www-unix.mcs.anl.gov/DSDP/`. `DSDP` is considered particularly efficient for solving problems where the solution is known to have low rank (as it is the case for Max-Cut instances), since it exploits low-rank structure and sparsity in the data. Further, `DSDP` has relatively low memory requirements for an interior-point method, and is indeed able to solve instances up to 10 000 nodes. We also include the Spectral Bundle method `SB` can be found in [17] and downloadable `http://www-user.tu-chemnitz.de/~ helmberg/`.

Among the NLP based methods, we choose as a term of comparison the code `SDPLR-MC`, proposed by Burer and Monteiro in [5] downloadable from the web page `http://dollar.biz.uiowa.edu/~ burer/software/SDPLR-MC`, and `EXPA` proposed in [15].

Both EXPA and SDPLR−MC have a structure similar to `SpeeDP`. Indeed, the main scheme differs in the way of finding a stationary point for problem (6).

For any fixed value of $r$, EXPA uses the nonmonotone Barzilai-Borwein gradient proposed in [16] (the same one used in `SpeeDP`) to minimize an exact penalty function for (6). SDPLR−MC uses an L-BFGS method to obtain a stationary point of (7). We remark again that `SDPLR-MC` does not certify global optimality of the produced solution in the sense that it does not check the global optimality condition $Q + \mathrm{Diag}(\lambda(\widehat{\mathrm{V}})) \succeq 0$, while both `EXPA` and `SpeeDP` do it.

We do not include in our comparison neither the code `SDPLR` defined in [5] nor the manifold optimization method `GenRTR` defined in [21]. Indeed the computational results in [5] show that `SDPLR-MC` outperforms `SDPLR` on the Max-Cut problem. Further, in [21] the authors report a comparison of a matlab implementation of `GenRTR` and `SDPLR` on some Max-Cut instances which are a subset of those used as benchmark set here and in [15]. Although the direct comparison in terms of computational time is not really fair, the authors stated that the two methods, `GenRTR` and `SDPLR`, may be considered to have comparable performances. Since `SDPLR` is always worse than `SDPLR-MC`, this implies that on the special structured problem (1) that comes out from Max-Cut, `GenRTR` should have worse performances than `SDPLR-MC`.

Our benchmark set consists in standard instances of the Max-Cut problem, with number of nodes ranging from 100 to 20 000, with different degrees of sparsity. The first set of problems belongs to the SDPLIB collection of semidefinite programming test problems (hosted by B. Borchers) that can be downloaded from the web page `http://infohost.nmt.edu/~sdplib`. The smallest problems (mcp set) have been contributed by Fujisawa [9], while the maxG problems were supplied by Benson [3]. The second set of problems belongs to the Gset of randomly generated problems by means of a machine-independent graph generator *rudy* [26]. These problems can also be downloaded from Burer's web page `http://dollar.biz.uiowa.edu/~burer/software/SDPLR`.

`SpeeDP, EXPA, SDPLR-MC`, and `SB` solve all the test problems, whereas `DSDP` runs out of memory on the two largest problems (G77 and G81 of the Gset collection). Hence we eliminate these two test problems in the comparison with `DSDP`.

We compare the different codes on the basis of the level of accuracy and of the computational time.

As for the accuracy, we report in Table 2 the primal and/or dual objective function value obtained by the five methods on all the instances. We note that `SDPLR-MC` only reports the primal objective value, while `SB` produces a value of the dual objective function that is a bound on the optimal value of problem (1). Similarly, for `SpeeDP` we report the primal objective function value, and the dual, where the dual is obtained by adding $n\lambda_{\min}(Q + \mathrm{Diag}(\lambda(\widehat{\mathrm{V}})))$ to the primal value whenever $\lambda_{\min}(Q + \mathrm{Diag}(\lambda(\widehat{\mathrm{V}})))$ is negative.

As regards the computational time, in order to have a better flavor of the results, we follow the approach proposed in [7] and we draw the cpu time performance profile of the five methods. To be more precise, let $p$ denote a particular problem and $s$ a particular solver. The idea is to compare the performance of solver $s$ on problem $p$ with the best performance by any solver on this particular

Table 2: Objective function values obtained by the three NLP based methods, SB and DSDP

| | SpeeDP | | EXPA | | SDPLR-MC | SB | DSDP | |
|---|---|---|---|---|---|---|---|---|
| | primal | dual obj | primal | dual | primal | dual | primal | dual |
| mcp100 | 226.157349 | 226.157303 | 226.15735 | 226.157394 | 226.15127 | 226.1592 | 226.15733 | 226.15735 |
| mcp124_1 | 141.990463 | 141.990494 | 141.99045 | 141.990402 | 141.99002 | 141.9937 | 141.99044 | 141.99048 |
| mcp124_2 | 269.880157 | 269.880890 | 269.88016 | 269.883087 | 269.88011 | 269.8822 | 269.88012 | 269.88017 |
| mcp124_3 | 467.750092 | 467.750092 | 467.75009 | 467.750092 | 467.75000 | 467.7537 | 467.75004 | 467.75012 |
| mcp124_4 | 864.411682 | 864.411682 | 864.41187 | 864.411926 | 864.41045 | 864.4156 | 864.41166 | 864.41187 |
| mcp250_1 | 317.264313 | 317.264313 | 317.26425 | 317.264313 | 317.26415 | 317.2708 | 317.26429 | 317.26435 |
| mcp250_2 | 531.930054 | 531.930115 | 531.92999 | 531.929993 | 531.92949 | 531.9349 | 531.92998 | 531.93009 |
| mcp250_3 | 981.172546 | 981.172607 | 981.17242 | 981.172424 | 981.17207 | 981.1780 | 981.17239 | 981.17257 |
| mcp250_4 | 1681.959595 | 1681.959595 | 1681.95960 | 1681.961426 | 1681.95581 | 1681.9750 | 1681.95995 | 1681.96011 |
| mcp500_1 | 598.148499 | 598.148499 | 598.14789 | 598.147888 | 598.14749 | 598.1588 | 598.14840 | 598.14852 |
| mcp500_2 | 1070.056152 | 1070.056152 | 1070.05615 | 1070.056152 | 1070.04501 | 1070.0759 | 1070.05660 | 1070.05677 |
| mcp500_3 | 1847.969971 | 1847.970093 | 1847.96936 | 1847.974731 | 1847.92925 | 1847.9836 | 1847.96947 | 1847.97003 |
| mcp500_4 | 3566.737549 | 3566.766357 | 3566.73779 | 3566.737793 | 3566.72863 | 3566.7479 | 3566.73765 | 3566.73806 |
| G01 | 12083.196289 | 12083.197266 | 12083.19727 | 12083.19727 | 12082.93730 | 12083.2650 | 12083.19640 | 12083.19770 |
| G60 | 15222.267578 | 15222.267578 | 15222.25488 | 15222.25488 | 15221.90950 | 15223.1930 | 15222.25710 | 15222.26810 |
| G11 | 629.146118 | 629.146118 | 629.14722 | 629.147217 | 629.15795 | 629.1701 | 629.16472 | 629.16478 |
| G14 | 3191.564209 | 3191.656982 | 3191.56567 | 3192.163574 | 3191.55893 | 3191.5847 | 3191.56609 | 3191.56681 |
| G22 | 14135.945312 | 14136.145508 | 14135.94336 | 14135.97754 | 14135.77440 | 14136.0440 | 14135.94470 | 14135.94570 |
| G32 | 1567.605591 | 1567.605591 | 1567.59265 | 1567.592651 | 1567.61532 | 1567.6519 | 1567.63942 | 1567.63965 |
| G35 | 8014.738770 | 8014.738770 | 8014.73584 | 8014.766602 | 8014.55729 | 8014.8070 | 8014.73758 | 8014.73972 |
| G36 | 8005.962891 | 8005.962891 | 8005.95850 | 8007.056152 | 8005.91627 | 8006.0213 | 8005.96316 | 8005.96379 |
| G43 | 7032.221680 | 7032.221680 | 7032.21973 | 7032.223145 | 7032.19023 | 7032.2749 | 7032.22079 | 7032.22185 |
| G48 | 5999.998047 | 6000.000000 | 5999.99561 | 5999.995605 | 5999.79596 | 6000.0000 | 5999.99852 | 6000.00000 |
| G51 | 4006.247559 | 4006.248291 | 4006.25391 | 4006.253906 | 4006.21749 | 4006.2745 | 4006.25460 | 4006.25553 |
| G52 | 4009.634766 | 4009.672363 | 4009.63696 | 4009.645264 | 4009.60292 | 4009.6574 | 4009.63834 | 4009.63877 |
| G55 | 11039.45898 | 11039.45898 | 11039.44922 | 11039.44922 | 11039.20140 | 11040.1590 | 11039.44910 | 11039.46050 |
| G57 | 3885.31787 | 3885.317871 | 3885.33887 | 3885.338867 | 3885.36524 | 3885.5197 | 3885.48675 | 3885.48917 |
| G58 | 20136.17383 | 20136.17383 | 20136.16992 | 20141.125 | 20135.59150 | 20136.2870 | 20136.18060 | 20136.18980 |
| G62 | 5430.506348 | 5430.506348 | 5430.79688 | 5430.796875 | 5430.72287 | 5430.9512 | 5430.90837 | 5430.91042 |
| G63 | 28244.353516 | 28244.458984 | 28244.30664 | 28260.37695 | 28243.63720 | 28244.5770 | 28244.40560 | 28244.41790 |
| G64 | 10465.827148 | 10469.678711 | 10465.87891 | 10472.86621 | 10465.82000 | 10465.9700 | 10465.89790 | 10465.90440 |
| G65 | 6205.268555 | 6205.268555 | 6205.32861 | 6205.328613 | 6205.28798 | 6205.5822 | 6205.53219 | 6205.53820 |
| G66 | 7076.871582 | 7076.871582 | 7076.92676 | 7076.926758 | 7076.93516 | 7077.2640 | 7077.20904 | 7077.21373 |
| G67 | 7744.033203 | 7744.033203 | 7744.14697 | 7744.146973 | 7744.06409 | 7744.4942 | 7744.42783 | 7744.43649 |
| G70 | 9861.485352 | 9861.485352 | 9861.48340 | 9861.483398 | 9861.24740 | 9861.7747 | 9861.51431 | 9861.52455 |
| G72 | 7808.015137 | 7808.015137 | 7808.04541 | 7808.04541 | 7808.16545 | 7808.5914 | 7808.53427 | 7808.53926 |
| G77 | 11044.382812 | 11044.983398 | 11045.36621 | 11045.36621 | 11045.08100 | 11045.7510 | **** | **** |
| G81 | 15655.122070 | 15655.122070 | 15655.44238 | 15655.44238 | 15655.12500 | 15656.2790 | **** | **** |

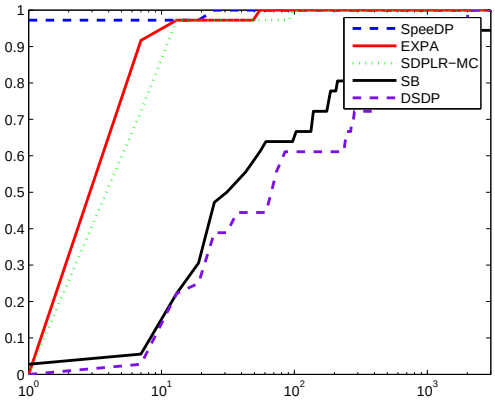


Figure 1: Comparison between NLP based methods, `SB` and `DSDP`

problem. To this aim, consider the performance ratio

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s'} : s' \in S\}},$$

where $t_{p,s}$ is the CPU time in seconds needed by solver $s$ to solve problem $p$. Given this performance ratio, a cumulative distribution function $\rho_s(\tau)$ is defined as:

$$\rho_s(\tau) = \frac{1}{n_p}\text{size}\{p \in P : r_{p,s} \leq \tau\}.$$

We draw $\rho_s(\tau)$ with respect to $\tau$, that is reported on the x-axis in a logarithmic scale.

In the picture, the higher the method the better, and the efficiency is measured by how fast the method reaches the value of 1 (since all the methods solve all the problems, all the methods reach the performance value 1 allowing a sufficiently large $\tau$).

In Figure 1 we compare all the five methods on the test problems solved by all of them (i.e. all the problems except G77 and G80).

In Figure 2, we report the comparison among the three low rank based methods and `SB` on all the test problems. It emerges from the profiles that `SpeeDP` outperforms the other methods.

Finally, we report the numerical results obtained by the heuristic described in Section 11 on some large random graphs. We used the graph generator *rudy* [26] to define instances with growing dimension and density and different weights. We first considered graphs with number of nodes $n$ equal to $500 + i \cdot 250$, for $i = 0, \ldots, 8$ and with edge *density* equal to $10\% + i \cdot 10\%$ for $i = 0, \ldots, 9$. For each pair ($n$, *density*) we generated three different graphs with positive weights ranging between 1 and 100. In Table 3 we report in each row the average values on the three problems in each class of CPU time, cut value, gap %, and the value

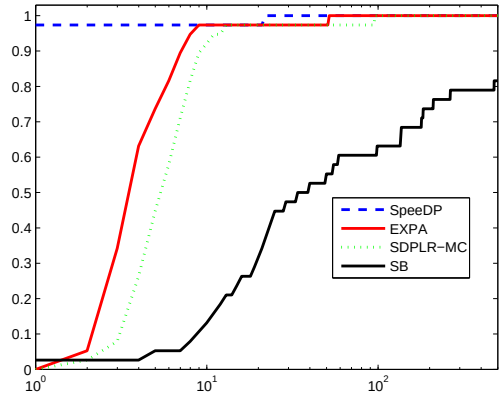


Figure 2: Comparison among the NLP based methods, and `SB`

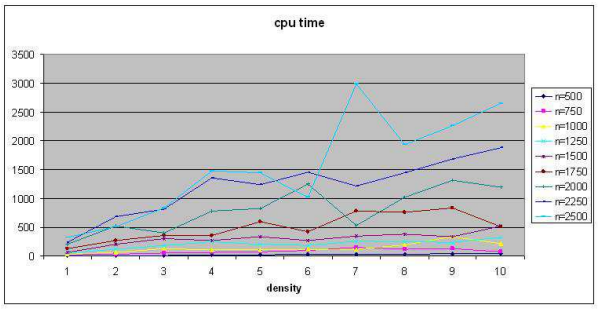


Figure 3: Average CPU time of the heuristic on the random graphs

0.87856×(upper bound), which represents the expected value of the Goemans-Williamson algorithm. We also draw in Figure 3 the average CPU time as a function of the density of the graph. As it emerges from the figure and the table, the heuristic is able to produce a good cut in a small amount of time, and as expected the performance of the heuristic is better on sparse graphs in term of time, but the gap decreases when the density of the graph increases.

Furthermore, we consider huge graphs, in order to verify how far we can go with the number of nodes. For this set of instances we run `SpeeDP` on a machine with 6G of RAM.

We generate three random graphs with 100 001 nodes, 7 050 827 edges and different weights. The results are in Table 4 where we report the ranges of the weights, the total time, the value of the bound, the best cut obtained and the % gap.

We also generated 6-regular graphs (3D toroidal grid graphs) with 1 030 301 nodes and 3 090 903 edges and different weights. The results are reported in

| | cut | time | gap % | 0.87856*ub |
|---|---|---|---|---|
| n=500 | | | | |
| | 388004.6667 | 6.216666667 | 4.595666667 | 356550.8369 |
| | 730297.3333 | 8.366666667 | 3.421133333 | 663560.8102 |
| | 1063049 | 13.4 | 2.761066667 | 959739.3467 |
| | 1391196.333 | 18.62666667 | 2.394766667 | 1251519.1163 |
| | 1714089 | 19.37666633 | 1.992666667 | 1535937.3951 |
| | 2032992.333 | 36.21333233 | 1.752933333 | 1817416.2683 |
| | 2347809 | 27.92000033 | 1.5559 | 2094784.6522 |
| | 2661106 | 30.53333433 | 1.295366667 | 2368226.2755 |
| | 2971604.333 | 44.373333 | 1.0805 | 2638941.8736 |
| | 3279198.333 | 47.47666667 | 0.880966667 | 2906354.3058 |
| n=750 | | | | |
| | 846281.6667 | 19.56000067 | 3.955333333 | 772917.3149 |
| | 1602525.667 | 34.23999967 | 3.0752 | 1451210.8624 |
| | 2344842.667 | 52.066667 | 2.407933333 | 2109690.0815 |
| | 3075792 | 62.92333367 | 2.038 | 2757341.3049 |
| | 3800724.667 | 71.54666633 | 1.735166667 | 3397102.7666 |
| | 4515431.333 | 100.419998 | 2.219533333 | 4055145.0852 |
| | 5223831.667 | 154.16333 | 1.951166667 | 4679018.8875 |
| | 5932074 | 117.4766693 | 2.158966667 | 5324216.9517 |
| | 6632193.667 | 126.8666653 | 1.421066667 | 5909605.1404 |
| | 7324080.333 | 81.35666933 | 1.5842 | 6536586.5558 |
| n=1000 | | | | |
| | 1470304.333 | 27.27666667 | 3.842166667 | 1341381.6741 |
| | 2804656.667 | 60.929999 | 2.8092 | 2533278.8535 |
| | 4116964 | 129.2566707 | 2.237433333 | 3697925.6641 |
| | 5407801.667 | 109.0300037 | 1.908166667 | 4841737.4244 |
| | 6694033 | 109.6433357 | 1.639166667 | 5977511.9713 |
| | 7971429.333 | 115.23333 | 1.9694 | 7141286.9576 |
| | 9234082.333 | 115.013331 | 1.6857 | 8249405.7535 |
| | 10487245 | 194.3833363 | 1.937533333 | 9392157.2383 |
| | 11727142.33 | 341.6733297 | 1.636033333 | 10471585.0468 |
| | 12962878.33 | 220.37 | 1.038566667 | 11506962.7354 |
| n=1250 | | | | |
| | 2261240.667 | 55.766665 | 3.555366667 | 2057267.4313 |
| | 4338898 | 117.859998 | 2.584166667 | 3910489.3026 |
| | 6376438.667 | 180.5166627 | 2.106733333 | 5720104.4340 |
| | 8396040.667 | 250.316666 | 1.7218 | 7503431.5860 |
| | 10395843 | 205.3633373 | 3.4118 | 9444982.7083 |
| | 12378616 | 179.5900063 | 1.239633333 | 11010166.9265 |
| | 14350605 | 256.8466697 | 2.522633333 | 12925918.8415 |
| | 16314387.67 | 247.2799987 | 1.7555 | 14584779.8125 |
| | 18264800 | 243.693334 | 1.495266667 | 16286651.5669 |
| | 20200056 | 321.730001 | 1.5469 | 18021482.4997 |
| n=1500 | | | | |
| | 3220711 | 60.03666533 | 3.394633333 | 2925643.823 |
| | 6195972 | 203.75 | 2.443333333 | 5576536.602 |
| | 9128485.333 | 305.3333283 | 1.932633333 | 8174914.701 |
| | 12020693.33 | 267.9600067 | 1.6445 | 10734573.2 |
| | 14899393 | 338.5299987 | 1.354266667 | 13267285.67 |
| | 17760397.33 | 272.16333 | 2.6834 | 16022282.32 |
| | 20603549.33 | 347.5099997 | 2.344366667 | 18525817.52 |
| | 23430844.67 | 384.0266673 | 2.016433333 | 21000489.11 |
| | 26236188 | 341.8233337 | 1.707533333 | 23443644.4 |
| | 29026448 | 521.1066593 | 1.414566667 | 25862214.06 |
| n=1750 | | | | |
| | 4344770.667 | 125.839999 | 3.216233333 | 3939910.081 |
| | 8382268 | 270.2400007 | 2.3461 | 7537097.712 |
| | 12355222.33 | 362.8666637 | 1.858433333 | 11056530.88 |
| | 16297722.33 | 356.3966677 | 2.1337 | 14624054.96 |
| | 20215156.67 | 602.439992 | 1.850566667 | 18088909.05 |
| | 24101639.33 | 422.2233273 | 2.4925 | 21702521.8 |
| | 27971634 | 777.3833313 | 1.7368 | 25001570.83 |
| | 31820130 | 760.8966573 | 1.881333333 | 28481837.5 |
| | 35640165.33 | 832.783315 | 1.591566667 | 31810367.49 |
| | 39437790.67 | 515.2300007 | 1.312266667 | 35103136.57 |
| n=2000 | | | | |
| | 5630775.667 | 208.1233367 | 3.072666667 | 5098977.742 |
| | 10892119.33 | 516.519989 | 2.176033333 | 9777614.603 |
| | 16076290 | 400.5799963 | 3.125133333 | 14565327.62 |
| | 21216458.67 | 778.7866413 | 2.036266667 | 19019499.13 |
| | 26329037.33 | 828.900004 | 2.1775 | 23635357.9 |
| | 31404458 | 1246.920024 | 1.915466667 | 28119172.62 |
| | 36443104 | 535.9733173 | 2.040133333 | 32670659.3 |
| | 41470418.67 | 1019.960002 | 1.777566667 | 37081879.77 |
| | 46475704 | 1320.056641 | 1.5193 | 41452038.69 |
| | 51442052 | 1198.696635 | 1.2375 | 45754212.3 |
| n=2250 | | | | |
| | 7092778.333 | 239.7566683 | 2.9204 | 6413414.787 |
| | 13724521.67 | 681.0033367 | 2.145733333 | 12316544.65 |
| | 20282490 | 817.880005 | 1.6331 | 18110391.01 |
| | 26779226 | 1363.850026 | 1.3677 | 23848933.49 |
| | 33236806 | 1234.75002 | 2.1361 | 29824281.87 |
| | 39660752 | 1457.619995 | 1.3802 | 35325275.19 |
| | 46049029.33 | 1214.609985 | 1.929166667 | 41237316.29 |
| | 52409853.33 | 1443.710001 | 1.646333333 | 46803264.48 |
| | 58740968 | 1682.113363 | 1.409033333 | 52334639.59 |
| | 65042482.67 | 1877.176676 | 1.171766667 | 57813304.6 |
| n=2500 | | | | |
| | 8707864.333 | 326.8133334 | 2.866133333 | 7869652.293 |
| | 16883800 | 512.99999 | 2.059233333 | 15138884.38 |
| | 24971508.67 | 834.2133483 | 2.199433333 | 22421516.56 |
| | 32984345.33 | 1475.053324 | 1.823933333 | 29507260.52 |
| | 40946184 | 1444.76001 | 2.447766667 | 36854243.7 |
| | 48873544 | 1019.210001 | 1.725333333 | 43679150.81 |
| | 56771676 | 2992.886719 | 1.486266667 | 50618557.71 |
| | 64627920 | 1933.013305 | 1.558366667 | 57664353.54 |
| | 72453770.67 | 2257.273397 | 1.352333333 | 64515830.64 |
| | 80230896 | 2650.269979 | 0.905633333 | 71126099.68 |



Table 3: Random graphs with weights in $[1, 100]$ and density from 10% to 100%

| Weights | Total CPU time | Upper Bound | Best Cut | gap% |
|---|---|---|---|---|
| 1 | 15 043.98 | 4 113 227.8 | 3 959 852 | 3.87 |
| [1, 100] | 15 142.22 | 212 076 831.2 | 203 236 495 | 4.35 |
| [−1000, 1000] | 15 919.40 | 21 006 071 437.9 | 20 129 935 523 | 4.35 |

Table 4: Random sparse graphs with 100 001 nodes and 7 050 827 edges

| Weights | Total CPU time | Upper Bound | Best Cut | gap% |
|---|---|---|---|---|
| 1 | 4 723 | 3 090 133 | 3 060 300 | 0.97 |
| [1, 10] | 22 042 | 15 454 739 | 15 338 007 | 0.76 |
| [1, 1000] | 29 072 | 1 545 550 679 | 1 534 441 294 | 0.72 |
| [−100, 100] | 47 491 | 57 288 795 | 49 111 079 | 14.27 |

Table 5: 6-regular graphs with 1 030 301 nodes and 3 090 903 edges

Table 5. To the best of our knowledge, no other methods can achieve this accuracy for graphs of this size.

# 13 Concluding Remarks and future works

In this paper, we define a fast globally convergent algorithm for solving problem (1), called `SpeeDP`, which falls in the low rank nonlinear programming approach. `SpeeDP` outperforms existing methods for solving the special structured semidefinite programming problem (1) and provides both a primal and a dual solution. We also define an heuristic to compute a cut which is an enhanced version of the Goemans-Williamson algorithm, and is suitable for graphs up to millions of nodes and edges. The heuristic provides both a feasible cut and a valid bound, hence hence it is able to provide a cut and a guaranteed bound on how much its weight deviates from the optimum. As a next step, we intend to include `SpeeDP` within a branch-and-bound scheme similarly to what has been done in the BiqMac code of [25], in such a way aiming at increasing the size of Max-Cut instances that can be solved exactly exploiting semidefinite programming.

# 14 Appendix: Technical proofs and results

*Proposition 4.* The first condition in (9) can be written in matrix form as:

$$\left(Q + \mathrm{Diag}(\widehat{\lambda})\right) \widehat{V} = 0 \tag{22}$$

where $\widehat{V}$ is the $n \times p$ matrix with rows $v_i^T$. Let $\widehat{V}$ be the $n \times r$ matrix satisfying (22) and the second order necessary condition

$$\left(Q + \mathrm{Diag}(\widehat{\lambda})\right) \bullet ZZ^T \geq 0 \qquad \text{for all } Z \in \mathbb{R}^{n\times r}: \quad E_{ii} \bullet \widehat{V}Z^T = 0 \;\; i = 1, \ldots, n.$$

Assume the $\text{rank}(\widehat{V}) = p < r$, namely $\widehat{V}$ is rank deficient, then there exists a $n \times p$ matrix $\widehat{V}_1$ such that

$$\widehat{V} = \widehat{V}_1 M^T \qquad M \in \mathbb{R}^{r\times p}.$$

Let $M_\perp \in \mathbb{R}^{r\times(r-p)}$ be a matrix such that

$$M^T M_\perp = 0 \quad M_\perp^T M_\perp = I_{r-p}$$

For any matrix $Z_1 \in \mathbb{R}^{n\times(r-p)}$, the matrix $Z = Z_1 M_\perp^T$ satisfies

$$E_{ii} \bullet \widehat{V} Z^T = E_{ii} \bullet \widehat{V}_1 M^T M_\perp Z_1^T = 0$$

so that we must have

$$\Big(Q + \text{Diag}(\widehat{\lambda})\Big) \bullet ZZ^T = \Big(Q + \text{Diag}(\widehat{\lambda})\Big) \bullet Z_1 Z_1^T \geq 0$$

for any $n \times (r-p)$ matrix $Z_1$, which is equivalent to $Q + \text{Diag}(\widehat{\lambda}) \succeq 0$, so that global optimality of $\widehat{V}$ follows from Theorem 3. If $r = n$ and $\text{rank}(\widehat{V}) < n$, the result follows from above. If instead $\text{rank}(\widehat{V}) = n$, the first order condition (22) implies $Q + \text{Diag}(\widehat{\lambda}) \equiv 0_{\text{n}\times\text{n}}$, so that global optimality of $\hat{v}$ follows from Theorem 3. □

We prove Proposition 6. We split it into two propositions.

**Proposition 15.** *For every $v \in S_\delta$ and for every given $\varepsilon > 0$, the following condition holds*

$$f_\varepsilon(v) \geq -C + \frac{1}{\varepsilon}\frac{(\|v_i\|^2 - 1)^2}{\delta^2}, \quad \textit{for all } i = 1, \ldots, n, \tag{23}$$

*where $C = \sum_{i=1}^n \sum_{j=1}^n |q_{ij}|$. Furthermore, for every given $\varepsilon > 0$ and for every given $v^0 \in S_\delta$, the level sets*

$$\mathscr{L}_\varepsilon(v^0) = \{v \in S_\delta : \ f_\varepsilon(v) \leq f_\varepsilon(v^0)\}$$

*of function $f_\varepsilon(v)$ are compact.*

*Proof.* First, for every $v$, we have that

$$\begin{aligned} f_r(v) &= \sum_{i=1}^n \sum_{j=1}^n q_{ij} \frac{v_i^T v_j}{\|v_i\|\|v_j\|} \geq -\sum_{i=1}^n \sum_{j=1}^n |q_{ij}| \frac{|v_i^T v_j|}{\|v_i\|\|v_j\|} \\ &\geq -\sum_{i=1}^n \sum_{j=1}^n |q_{ij}| \frac{\|v_i\|\|v_j\|}{\|v_i\|\|v_j\|} = -C. \end{aligned}$$

Hence, (23) follows from simple majorizations. Now, we prove boundedness of $\mathscr{L}_\varepsilon(v^0)$. Let $\{v^k\} \in \mathscr{L}_\varepsilon(v^0)$ be a sequence of points such that $\|v^k\| \to \infty$. Assume without loss of generality that $\|v_1^k\| \to \infty$. By using (23), we can write:

$$f_\varepsilon(v^k) \geq -C + \frac{1}{\varepsilon}\frac{(\|v_1^k\|^2 - 1)^2}{\delta^2},$$

so that $f_\varepsilon(v)$ is coercive and the level set is bounded. On the other hand, any limit point of a sequence cannot belong to the boundary of $S_\delta$. Indeed, if $\|\hat{v}_i\|^2 = 1 - \delta$ for some $i$, then (13) implies $d(\hat{v}_i) = 0$, and hence

$$\lim_{k\to\infty} f_\varepsilon(v^k) = \infty,$$

but this contradicts $v^k \in \mathscr{L}_\varepsilon(v^0)$ for $k$ sufficiently large. Therefore the level set $\mathscr{L}_\varepsilon(v^0)$ is also closed, and the thesis follows. □

Next proposition gives a bound on the value of $\|v_i\|$ for all $i = 1, \ldots, n$ in the level set.

**Proposition 16.** *Let $\varepsilon > 0$ and $v^0 \in \mathscr{F}$. Then, we have*

$$\mathscr{L}_\varepsilon(v^0) \subseteq \left\{ v \in \mathbb{R}^{nr} : \|v_i\|^2 \leq (2C\varepsilon\delta^2)^{\frac{1}{2}} + 1, \quad i = 1, \ldots, n \right\}.$$

*Proof.* For any given $v \in \mathscr{L}_\varepsilon(v^0)$, because $v^0 \in \mathscr{F}$, we can write

$$f_\varepsilon(v) \leq f_\varepsilon(v^0) = f_r(v^0) \leq C,$$

where $C$ is defined in Proposition 6. Moreover, using (23), we have

$$f_\varepsilon(v) \geq -C + \frac{1}{\varepsilon}\frac{(\|v_j\|^2 - 1)^2}{\delta^2}, \quad j = 1, \ldots, n,$$

so that

$$\|v_j\|^2 \leq (2C\varepsilon\delta^2)^{\frac{1}{2}} + 1, \quad j = 1, \ldots, n.$$

□

*Proposition 10.* By (21), for a fixed value $\varepsilon > 0$ the sequence $\{v^k\}$ stays in the compact level set $\mathscr{L}_\varepsilon(v^0)$. The proof is by induction. Assume that there exists $\bar{\varepsilon} > 0$ such that, for any $\varepsilon \geq \bar{\varepsilon}$, it is true that $\|v_i^k\|^2 \geq 1$. We show that is true also for $k + 1$. We can write

$$\begin{aligned}
\|v_i^{k+1}\|^2 &= \|v_i^k\|^2 + (\alpha^k)^2\|\nabla_{v_i} f_\varepsilon(v^k)\|^2 - 2\alpha^k (v_i^k)^T \nabla_{v_i} f_\varepsilon(v^k) \\
&= \|v_i^k\|^2 + (\alpha^k)^2\|\nabla_{v_i} f_\varepsilon(v^k)\|^2 - \frac{8\alpha^k}{\varepsilon}\frac{(\|v_i^k\|^2 - 1)\|v_i^k\|^2}{\delta^2} \\
&\geq \|v_i^k\|^2 - \frac{8\alpha_M}{\varepsilon\delta^2}(\|v_i^k\|^2 - 1)\|v_i^k\|^2,
\end{aligned}$$

where the second equality derives from (16), keeping in mind that $\|v_i^k\| \geq 1$. If $\|v_i^k\| = 1$, then $\|v_i^{k+1}\|^2 \geq 1$. Otherwise, if $\|v_i^k\| > 1$, we need to verify that a value of $\bar{\varepsilon}$ exists such that for all $\varepsilon \geq \bar{\varepsilon}$

$$(\|v_i^k\|^2 - 1) - \frac{8\alpha_M}{\varepsilon\delta^2}(\|v_i^k\|^2 - 1)\|v_i^k\|^2 \geq 0,$$

namely

$$1 - \frac{8\alpha_M}{\varepsilon\delta^2}\|v_i^k\|^2 \geq 0. \tag{24}$$

By Proposition 16, we have that for all $k$

$$\|v_i^k\|^2 \leq (2C\varepsilon\delta^2)^{\frac{1}{2}} + 1 \quad i = 1, \ldots, n. \tag{25}$$

Therefore (25) combined with (24) implies

$$\varepsilon - 8\frac{\alpha_M}{\delta^2}\left((2C\delta\varepsilon)^{\frac{1}{2}} + 1\right) \geq 0$$

which is satisfied for all $\varepsilon \geq \overline{\varepsilon}$. □

# References


[1] P.-A. Absil, C. Baker, and K. Gallivan, *Trust-region methods on Riemannian manifolds*, Journal Foundations of Computational Mathematics, 7 (2007), pp. 303–330.

[2] A. Barvinok, *Problems of distance geometry and convex properties of quadratic maps*, Discrete Computational Geometry, 13 (1995), pp. 189–202.

[3] S. J. Benson, Y. Ye, and X. Zhang, *Solving large-scale sparse semidefinite programs for combinatorial optimization*, SIAM Journal on Optimization, 10 (2000), pp. 443–461.

[4] D. P. Bertsekas, *Nonlinear Programming*, Athena Scientific, 1999.

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

[6] C. Delorme and S. Poljak, *Laplacian eigenvalues and the maximum cut problem*, Mathematical Programming, 62 (1993), pp. 557–574.

[7] E. Dolan and J. Morè, *Benchmarking optimization software with performance profile*, Mathematical Programming, Ser. A, 91 (2002), pp. 201–213.

[8] I. Fischer, G. Gruber, F. Rendl, and R. Sotirov, *Computational experience with a bundle approach for semidefinite cutting plane relaxations of Max-Cut and equipartition*, Math. Program., 105 (2006), pp. 451–469.

[9] K. Fujisawa, M. Fukuda, M. Kojima, and K. Nakata, *Numerical evaluation of sdpa (semidefinite programming algorithm)*, in High Performance Optimization, H.Frenk, K. Roos, T. Terlaky, and S. Zhang, eds., Kluwer Academic Press, 1999, pp. 267–301.

[10] M. Goemans and D. Williamson, *Improved approximation algorithms for Maximum Cut and satisfiability problems using Semidefinite Programming*, Journal of the ACM, 42 (1995), pp. 1115–1145.

[11] M. X. Goemans and D. P. Williamson, *.878-approximation algorithms for max cut and max 2sat*, in Proceedings of the Twenty-Sixth Annual ACM Symposium on the Theory of Computing, Montreal, Quebec, Canada, 1994, pp. 422–431.

[12] ———, *Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming*, J. Assoc. Comput. Mach., 42 (1995), pp. 1115–1145. preliminary version see [11].

[13] L. Grippo, L. Palagi, M. Piacentini, and V. Piccialli, *An unconstrained approach for solving low rank SDP relaxations of* $\{-1, 1\}$ *quadratic problems*, Tech. Report 1.13, Dip. di Informatica e sistemistica A. Ruberti, Sapienza Università di Roma, 2009.

[14] L. Grippo, L. Palagi, and V. Piccialli, *Necessary and sufficient global optimality conditions for NLP reformulations of linear SDP problems*, Journal of Global Optimization, 44 (2009), pp. 339–348.

[15] ———, *An unconstrained minimization method for solving low rank SDP relaxations of the Max Cut problem*, Mathematical Programming, (2010). DOI: 10.1007/s10107-009-0275-8.

[16] L. Grippo and M. Sciandrone, *Nonmonotone globalization techniques for the Barzilai-Borwein gradient method*, Computational Optimization and Applications, 23 (2002), pp. 143–169.

[17] C. Helmberg and F. Rendl, *A spectral bundle method for semidefinite programming*, SIAM Journal on Optimization, 10 (2000), pp. 673–696.

[18] S. Homer and M. Peinado, *Desing and performance of parallel and distributed approximation algorithm for the Maxcut*, Journal of Parallel and Distributed Computing, 46 (1997), pp. 48–61.

[19] M. Laurent, S. Poljak, and F. Rendl, *Connections between semidefinite relaxations of the max-cut and stable set problems*, Mathematical Programming, 77 (1997), pp. 225–246.

[20] F. Liers, M. Jünger, G. Reinelt, and G. Rinaldi, *Computing exact ground states of hard Ising spin glass problems by branch-and-cut*, in New Optimization Algorithms in Physics, A. Hartmann and H. Rieger, eds., Wiley-VCH Verlag, 2004, pp. 47–69.

[21] F. B. M. Journée, P. Absil, and R. Sepulchre, *Low-rank optimization for semidefinite convex problems*, Tech. Report arXiv:0807.4423v1, accepted for publication in SIAM Journal on Optimization, 2010., 2008.

[22] G. Pataki, *On the rank of extreme matrices in semidefinite programs and the multiplicity of optimal eigenvalues*, Mathematics of Operations Research, 23 (1998), pp. 339 –358.

[23] S. Poljak and F. Rendl, *Solving the Max-Cut problem using eigenvalues*, Discrete Applied Mathematics, 62 (1995), pp. 249–278.

[24] G. R., P. S., and W. W., *Extremal Correlation Matrices*, Linear Algebra Application, 134 (1990), pp. 63–70.

[25] F. Rendl, G. Rinaldi, and A. Wiegele, *Solving Max-Cut to optimality by intersecting semidefinite and polyhedral relaxations*, Mathematical Programming, 121 (2010), pp. 1436–4646.

[26] G. Rinaldi, *Rudy-graph generator*.
http://www-user.tu-chemnitz.de/∼helmberg/sdp_software.html, 1998.

[27] H. W. S. Poljak, F. Rendl, *A recipe for semidefinite relaxation for 0-1 quadratic programming*, Journal of Global Optimization, 7 (1995), pp. 51–73.