# SpeeDP: an algorithm to compute SDP bounds for very large Max-Cut instances

**Luigi Grippo · Laura Palagi · Mauro Piacentini · Veronica Piccialli · Giovanni Rinaldi**

**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 non-convex 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. We further include SpeeDP within a simply modified version of the Goemans–Williamson algorithm and we show that

L. Grippo · L. Palagi · M. Piacentini
Dipartimento di Ingegneria informatica automatica e gestionale A. Ruberti,
Sapienza Università di Roma, via Ariosto, 25, 00185 Rome, Italy
e-mail: grippo@dis.uniroma1.it

L. Palagi
e-mail: laura.palagi@uniroma1.it

M. Piacentini
e-mail: piacentini@dis.uniroma1.it

V. Piccialli (✉)
Dipartimento di Ingegneria Civile e Ingegneria Informatica, Università degli Studi di Roma Tor Vergata,
via del Politecnico, 1, 00133 Rome, Italy
e-mail: piccialli@disp.uniroma2.it

G. Rinaldi
Istituto di Analisi dei Sistemi ed Informatica "A. Ruberti" del CNR,
viale Manzoni, 30, 00185 Rome, Italy
e-mail: rinaldi@iasi.cnr.it

the corresponding heuristic `SpeeDP-MC` can generate high-quality cuts for very large, sparse graphs of size up to a million nodes in a few hours.

**Keywords** Semidefinite programming · Low rank factorization · Unconstrained binary quadratic programming · Max-Cut · Nonlinear programming

**Mathematics Subject Classification (1991)** 20E28 · 20G40 · 20C20

## 1 Introduction

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

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

where $Q \in \mathscr{S}^n$ is given, $X \in \mathscr{S}^n$ ($\mathscr{S}^n$ being the space of the $n \times n$ real symmetric matrices), the symbol '$\bullet$' denotes the trace-inner product, $e \in \mathbb{R}^n$ is the vector of all ones, $\text{diag}(X)$ stands for the $n$-vector corresponding to the main diagonal of $X$, and, finally, the constraint $X \succeq 0$ requires $X$ to be positive semidefinite.

Semidefinite programming problems of this form arise as relaxations of unconstrained $\{-1, 1\}$ quadratic problems (see, e.g., [8,12,23,28]):

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

which are equivalent to the Max-Cut problem.

Given the weighted adjacency matrix $A$ of a weighted graph $G = (N, E)$, the Max-Cut problem calls for a bipartition $(S, N \backslash S)$ of its nodes so that the weight of the corresponding cut, i.e., of the edges that go across the two sets of the bipartition, is maximized. Denote by $L$ the Laplacian matrix associated with $A$ and defined by $L := \text{diag}(Ae) - A$. Represent each bipartition $(S, N \backslash S)$ by an $n$-vector defined by $x_i = 1$ for $i \in S$ and $x_i = -1$ for $i \notin S$. 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 $\text{diag}(x x^T) = e$, it is known that problem (1) with $Q = -\frac{1}{4} L$ provides a relaxation of Max-Cut. Producing a solution to problem (1) efficiently is then of great interest for solving the corresponding integer problem (2) exactly or for designing good heuristics (see, e.g., [30,32]).

The aim of this paper is twofold: on one side we define an efficient algorithm for solving large scale instances of problem (1); on the other, by exploiting this useful tool, we make it possible to apply a Goemans–Williamson type heuristic to find good feasible solutions to very large instances of Max-Cut.

It is well known that problem (1) can be solved by semidefinite interior point methods. However, these approaches typically become less attractive for instances $n > 20000$ because those methods that make use of an explicit representation of the basis matrix $X$ have excessive memory requirements, while those that are based on iterative methods (and usually solve the dual of problem (1)) are, to the best of our knowledge, still too slow in practice. These limitations have motivated searching for methods that are less demanding in terms of memory allocation and of computational cost. In this perspective, the constraint structure of problem (1) has been exploited in the literature to define special purpose algorithms.

One possibility is to reformulate the dual to (1) as an eigenvalue optimization problem that can be solved, e.g., by spectral bundle methods [19].

The other option is to use nonlinear programming reformulations that eliminate the semidefiniteness constraint from the primal problem (1) (see, e.g., [2,5–7,11,16, 20,21]). This 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 an $n \times r$ real matrix. Therefore, the positive semidefiniteness constraint can be eliminated and problem (1) reduces to the low rank SDP (LRSDP) formulation

$$\min_{V \in \mathbb{R}^{n \times r}} \left\{ Q \bullet VV^T : \operatorname{diag}(VV^T) = e \right\}. \tag{4}$$

Problem (4) can be written as the nonlinear programming (NLP) problem

$$\min_{v \in \mathbb{R}^{nr}} \left\{ q_r(v) = \sum_{i=1}^{n} \sum_{j=1}^{n} q_{ij} v_i^T v_j : \|v_i\|^2 = 1, \quad i = 1, \ldots, n \right\}, \tag{5}$$

where $v_i$, $i = 1, \ldots, n$, with $r \leq n$, are the columns of the matrix $V^T$ and $v \in \mathbb{R}^{nr}$ is the vector obtained by stacking the column vectors $v_i$ for $i = 1, \ldots, n$. Indeed this relaxation was first derived by Goemans and Williamson in [12] by replacing each variable $x_i$ of problem (2) with a vector $v_i \in \mathbb{R}^r$. It is possible to state conditions that ensure correspondence among global solutions of problem (5) and solutions of problem (1). Moreover, although problem (5) is non-convex, it is also possible to state necessary and sufficient optimality conditions that can be used to check global optimality [15,16,21].

In most of the papers based on solving problem (5), the solution is achieved by means of an unconstrained reformulation. Indeed, the first idea of an unconstrained model for problem (1) goes back to Homer and Peinado [20], but the dimension of the resulting problem makes the method prohibitive for large scale problems. Burer and Monteiro in [7] introduced an augmented Lagrangian for solving the low rank reformulation of general linearly constrained SDP problems. In the numerical results section of [7], for the special case (4) arising from Max-Cut, they combine the Homer and Peinado approach 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 $r \ll n$, they get the unconstrained program

$$\min_{v \in \mathbb{R}^{nr}} \left\{ \phi(v) = \sum_{i=1}^{n} \sum_{j=1}^{n} q_{ij} \frac{v_i^T v_j}{\|v_i\| \|v_j\|}, \quad v_i \in \mathbb{R}^r \right\}. \tag{6}$$

In [16], the constraints structure in problem (5) is exploited to get an expression of the Lagrange multipliers $\lambda_i$, for $i = 1, \ldots, n$, in closed form as a function of the variables $v$ (see (10)). Using such an expression in the augmented Lagrangian function for problem (5) and choosing a sufficiently large value $\eta > 0$ for the penalty parameter, the authors obtain an exact penalty function of the form

$$P(v) = \sum_{i=1}^{n} \sum_{j=1}^{n} q_{ij} v_i^T v_j + \sum_{i=1}^{n} \lambda_i(v) \left( \|v_i\|^2 - 1 \right) + \frac{1}{2\eta} \sum_{i=1}^{n} \left( \|v_i\|^2 - 1 \right)^2. \tag{7}$$

A single unconstrained minimization of the twice continuously differentiable function $P$ is enough to find a stationary point of problem (5). Computational experiments in [16] with the resulting algorithmic scheme, called EXPA, showed significant computational advantages with respect to the best codes available in literature.

In this paper, we start from (6) to get a new unconstrained problem, for which we retain a complete equivalence with problem (1). The specific feature of this formulation is that we add a regularization term to the function $\phi$ in (6), which ensures compactness of the level sets of the new merit function. This allows us to use standard optimization algorithms to solve problem (5). As we discuss in Sect. 4, this new unconstrained approach has some advantages also over (7), which are confirmed by the numerical results reported in Sect. 6. Indeed, the new algorithmic scheme, called SpeeDP, outperforms the best existing methods for solving problem (1). Furthermore, SpeeDP (like other similar low rank approaches) has some nice features that can be exploited to design a Max-Cut heuristic to find good cuts for very large graphs. Indeed, if the value of $r$ is fixed to a value independent of $n$ (and if, as it is reasonable to assume for large sparse instances, the graph has $\mathcal{O}(n)$ edges), SpeeDP is able to derive a valid lower bound for problem (1), requiring $\mathcal{O}(n)$ memory. In addition, it outputs the Gramian 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 to find a good cut, essentially without any additional computational effort. Therefore, it is possible to produce also an upper bound (close to the SDP bound) for very large instances of problem (2). These features altogether make it possible to find cuts in sparse graphs with millions of nodes and edges, with observed gap 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 these algorithms provide a bound on the optimality error for the generated solutions that is close enough to the optimum to be of practical use.[1] Excluding the heuristics with a certified a priori bound (like the one of Goemans and Williamson)

---

[1] Actually, simple bounds like, e.g., the sum of all positive edge weights, can always be easily provided, but they are evidently useless to evaluate the quality of a heuristic solution.

the only cases when this bound is computed are those of the exact algorithms that compute an upper bound on the value of an optimal solution, by solving a relaxation of problem (3). If these algorithms are interrupted at an early stage, they provide a heuristic cut, as a side product, along with an upper bound on the optimal value. However, the computational studies based on this type of exact algorithms consider graphs much smaller than those used for the test bed of this paper (see, e.g., [24] for the polyhedral relaxations and [32] for a combination of SDP and polyhedral relaxation). Despite the fact that we proposed a slight modification of the Goemans and Williamson algorithm, our primary intent is not to contribute to the algorithms for Max-Cut with a new heuristic but rather to make it feasible to compute an SDP based upper bound for very large instances.

Before concluding this section, we note that other relaxations for problem (2) have been proposed in literature. Among these, besides the above mentioned polyhedral relaxations, we mention the second order cone (SOC) approximation (see, e.g., [22,26]). Computational experiments based on SOC relaxation consider much smaller instances than the ones considered in this paper (see also the computational results in [32]). However, to the best of our knowledge, a comparison between SOCP and low rank SDP approaches, when embedded in a branch and bound procedure, has not been performed yet and would require further research work.

The paper is structured as follows: in Sect. 2 we report on some known results about the LRSDP problem (4). In Sect. 3 we define the new unconstrained reformulation, while in Sect. 4 we formally define the algorithm SpeeDP. In Sect. 5 we define our heuristic, SpeeDP-MC, for finding good feasible solutions to large sparse instances of Max-Cut. In Sect. 6 we compare the performance of SpeeDP against other existing approaches for the SDP problem (1) on an extended set of instances of the Max-Cut problem. Furthermore, we use SpeeDP-MC to find good solutions to large and huge Max-Cut instances from random graphs.

## 2 Key results on the low rank SDP formulation and notation

In this section we define our basic notation and we report some useful results on the LRSDP formulation (4).

Throughout the paper, for an $r \times n$ matrix $M$, by $\text{vec}(M) \in \mathbb{R}^{rn}$ we denote the operator that creates a column vector by stacking the column vectors of $M$. Given a vector $v \in \mathbb{R}^m$, we define $B_\rho(v) = \{y \in \mathbb{R}^m : \|y - v\| < \rho\}$, for some $\rho > 0$. For a given scalar $x$, by $(x)_+$ we denote the maximum between $x$ and zero, namely $(x)_+ = \max(x, 0)$.

A global minimum point of (4) is a solution to problem (1) provided that

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

where $\mathscr{X}^*_{\text{SDP}}$ 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 [1,18,29], that gives

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

Let $v \in \mathbb{R}^{nr}$ be the vector obtained by stacking the column vectors $v_i$ for $i = 1, \ldots, n$. We say that *a point $v^* \in \mathbb{R}^{nr}$ solves problem* (1) if $X^* = V^* V^{*T}$ is an optimal solution to problem (1) (where $\text{vec}(V^{*T}) = v^*$). This implies, by definition, that $r \geq r_{\min}$.

The Karush–Kuhn–Tucker (KKT) conditions for problem (5) are written as follows: for some $\lambda \in \mathbb{R}^n$

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

We define a stationary point of problem (5) to be a $\hat{v} \in \mathbb{R}^{nr}$ satisfying (9) with a suitable multiplier $\hat{\lambda} \in \mathbb{R}^n$. Given a local minimizer $\hat{v} \in \mathbb{R}^{nr}$ of problem (5), the KKT conditions are necessary for optimality and there exists a unique $\hat{\lambda} \in \mathbb{R}^n$ such that $(\hat{v}, \hat{\lambda})$ satisfies (9). Indeed, given a pair $(v, \lambda)$ satisfying the conditions (9), the multiplier $\lambda$ can be expressed uniquely as a function of $v$, namely

$$\lambda_i(v) = -v_i^T \sum_{j=1}^{n} q_{ij} v_j, \quad i = 1, \ldots, n. \tag{10}$$

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

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

Although problem (5) is non-convex, the primal-dual optimality conditions for (1), combined with the necessary optimality conditions for (5), lead to the global optimality condition proved in [15,16] and stated in the next proposition.

**Proposition 1** *A point $v^* \in \mathbb{R}^{nr}$ is a global minimizer of problem* (5) *that solves problem* (1) *if and only if it is a stationary point of problem* (5) *and satisfies*

$$Q + \text{Diag}(\lambda(v^*)) \succeq 0,$$

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

Thanks to the above proposition, given a stationary point of problem (5), we can prove its optimality by 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 problem (1) as a special case. It is proved that a local minimizer $\widehat{V} \in \mathbb{R}^{n \times r}$ 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 $\text{rank}(\widehat{V}) < r$. Actually,

looking at the proof, it turns out that the assumption that $\widehat{V}$ is a local minimizer can be relaxed. Instead, it is enough to require that $\widehat{V}$ satisfies the second order necessary conditions for the LRSDP problem (see [14] for the restatement and the proof of this condition).

## 3 A new unconstrained formulation of the SDP problem

We start with the unconstrained formulation (6) proposed in [7], where it was shown to be quite effective in computations. It is easy to show that problems (4) and (6) are equivalent (see [13]). However, function $\phi$ in (6) is not defined at those points where $\|v_i\| = 0$ for at least one index $i$, and it has non compact level sets, so that standard convergence results for unconstrained minimization methods are not immediately applicable.

We modify the objective function $\phi$ given in (6) in order to obtain an unconstrained problem that can be solved by standard methods. In particular, we add the regularization term

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

where the term

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

acts as a shifted barrier on the open set

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

For a fixed $\varepsilon > 0$ and for a fixed $r \geq 1$, we consider the unconstrained minimization problem

$$\min_{v \in \mathbb{R}^{nr}} \left\{ f_\varepsilon(v) = \phi(v) + \frac{1}{\varepsilon} \sum_{i=1}^{n} \frac{(\|v_i\|^2 - 1)^2}{d(v_i)}, \quad v \in S_\delta \right\}. \tag{14}$$

Both $\phi$ and $S_\delta$ depend on $r$; however, we omit the explicit indication of this dependency to simplify notation.

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

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

where

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

The first important property is the compactness of the level sets of function $f_\varepsilon$, which guarantees the existence of a solution to problem (14). In the following, $\mathscr{F}$ stands for the feasible set of problem (5), i.e.,

$$\mathscr{F} = \{v \in \mathbb{R}^{nr} : \|v_i\|^2 = 1, \quad i = 1, \ldots, n\}.$$

**Proposition 2** *For every given $\varepsilon > 0$ and for every given $v^0 \in \mathscr{F}$, the level set $\mathscr{L}_\varepsilon(v^0) = \{v \in S_\delta : f_\varepsilon(v) \le f_\varepsilon(v^0)\}$ is compact and*

$$\mathscr{L}_\varepsilon(v^0) \subseteq \left\{v \in \mathbb{R}^{nr} : \|v_i\|^2 \le (2\overline{C}\varepsilon\delta^2)^{\frac{1}{2}} + 1, \quad i = 1, \ldots, n\right\}, \tag{15}$$

*with $\overline{C} = \sum_{i=1}^{n}\sum_{j=1}^{n}|q_{ij}| > 0$.*

*Proof* First, for every $v \in \mathbb{R}^{nr}$, with $v_i \neq 0$ for $i = 1, \ldots, n$, we have that

$$\phi(v) = \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij}\frac{v_i^T v_j}{\|v_i\|\|v_j\|} \ge -\sum_{i=1}^{n}\sum_{j=1}^{n}|q_{ij}|\frac{|v_i^T v_j|}{\|v_i\|\|v_j\|} \ge -\overline{C}.$$

Hence, we get

$$f_\varepsilon(v) \ge -\overline{C} + \frac{1}{\varepsilon}\frac{(\|v_i\|^2 - 1)^2}{\delta^2}, \quad \text{for all } i = 1, \ldots, n. \tag{16}$$

For every given $v \in \mathscr{L}_\varepsilon(v^0)$, as $v^0 \in \mathscr{F}$, we can write $f_\varepsilon(v) \le f_\varepsilon(v^0) = \phi(v^0) \le \overline{C}$, so that, using (16), we get $\|v_i\|^2 \le (2\overline{C}\varepsilon\delta^2)^{\frac{1}{2}} + 1, i = 1, \ldots, n$. This implies that (15) holds and hence that $\mathscr{L}_\varepsilon(v^0)$ is bounded.

On the other hand, any limit point $\hat{v}$ of a sequence of points $\{v^k\}$ in $\mathscr{L}_\varepsilon(v^0)$ 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 claim follows. □

The following theorem states the correspondence between stationary points, local/global minimizers of (14), and stationary points, local/global minimizers of (5), respectively.

This result exploits an interesting property of the objective function $\phi$ of problem (6). For every $v \in S_\delta$, its gradient with respect to $v_i$ is orthogonal to the vector $v_i$, namely the following holds, for $i = 1, \ldots, n$:

$$v_i^T \nabla_{v_i}\phi(v) = 2\left[\sum_{j=1}^{n} q_{ij}\left(\frac{v_i^T}{\|v_i\|} - \frac{v_i^T v_i}{\|v_i\|^2}\frac{v_i^T}{\|v_i\|}\right)\frac{v_j}{\|v_j\|}\right] = 0. \tag{17}$$

**Theorem 1** *For every $\varepsilon > 0$ and $r \geq 1$, $\hat{v}$ is a stationary point, a local/global minimizer of* (14) *in $S_\delta$ if and only if it is a stationary point, a local/global minimizer of* (5)*, respectively.*

*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 $\nabla_{v_i} f_\varepsilon(v)$ and by (17), we get, for every $v_i$ with $i = 1, \ldots, n$,

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

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

$$v_i^T \nabla_{v_i} f_\varepsilon(v) = \frac{4}{\varepsilon} \frac{(\|v_i\|^2 - 1)\|v_i\|^2}{\delta^2}, \tag{18}$$

otherwise

$$v_i^T \nabla_{v_i} f_\varepsilon(v) = \frac{4}{\varepsilon} \frac{(\|v_i\|^2 - 1)\|v_i\|^2}{d(v_i)} \left(1 + \frac{(\|v_i\|^2 - 1)^2}{d(v_i)}\right). \tag{19}$$

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

$$f_\varepsilon(v) = q_r(v) \tag{20}$$

$$\nabla_{v_i} f_\varepsilon(v) = 2 \sum_{j=1}^{n} q_{ij}(I_r - v_i v_i^T)v_j, \quad i = 1, \ldots, n. \tag{21}$$

Now we prove the correspondences stated in our claims.

*Case 1. (Correspondence of stationary points).*
    *Necessity.* By (18) and (19), with $\hat{v} \in S_\delta$ a stationary point of $f_\varepsilon$, we have $\hat{v} \in \mathscr{F}$. Hence, as a result of (21) and (11), $\hat{v}$ is a stationary point also for problem (5).
    *Sufficiency.* Let $\hat{v}$ be a stationary point for problem (5). Then, $\hat{v} \in \mathscr{F}$ and (11) holds, so that from (21) we get $\nabla_{v_i} f_\varepsilon(\hat{v}) = 0$ for $i = 1, \ldots, n$.

*Case 2. (Correspondence of global minimizers).*
    *Necessity.* By Proposition 2, 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 (5). Then there exists a point $v^*$, global minimizer of problem (5), 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 $f_\varepsilon$.
    *Sufficiency.* The claim is true by similar arguments.

*Case 3. (Correspondence of local minimizers).*
    *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, $f_\varepsilon(\hat{v}) = q_r(\hat{v})$. Furthermore, there exists a $\rho > 0$ such that $q_r(\hat{v}) =$

$f_\varepsilon(\hat{v}) \leq f_\varepsilon(v)$, for all $v \in S_\delta \cap B_\rho(\hat{v})$. Therefore, by using (20), for all $v \in \mathscr{F} \cap B_\rho(\hat{v})$, we have $q_r(\hat{v}) \leq f_\varepsilon(v) = q_r(v)$ and hence $\hat{v}$ is a local minimizer for problem (5).

*Sufficiency.* Since $\hat{v}$ is a local minimizer of (5), 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 $p(v) \in \mathscr{F} \cap B_\rho(\hat{v})$, where $p(v)$ has components $p_i(v) = v_i/\|v_i\|$ for $i = 1, \ldots, n$. Indeed, in this case we have $q_r(\hat{v}) = f_\varepsilon(\hat{v}) \leq q_r(p(v)) = f_\varepsilon(p(v)) \leq f_\varepsilon(v)$. Given $v_i \neq 0 \in \mathbb{R}^r$, its projection over the unit norm set is simply $v_i/\|v_i\|$, so that for $\hat{v} \in \mathscr{F}$ we have $\| v_i - \hat{v}_i \| \geq \| v_i - v_i/\| v_i \| \|$. Hence, for a chosen $\gamma < \rho/2$, we can write

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

Therefore, we have $f_\varepsilon(\hat{v}) \leq f_\varepsilon(v)$ for all $v \in S_\delta \cap B_\gamma(\hat{v})$, so that $\hat{v}$ is a local minimizer also for (14). □

## 4 `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 Sect. 2 we have seen that for $r \geq r_{\min}$ a global solution of problem (5) provides a solution to problem (1). Moreover, Theorem 1 states a complete correspondence between problems (5) and (14). Since $f_\varepsilon$ is continuously differentiable over the set $S_\delta$ and, by Proposition 2, it has compact level sets, we can find a stationary point of problem (14) by applying any globally convergent unconstrained minimization method (see, e.g., [4]).

The value of $r_{\min}$ is not known a priori and, in principle, the only computable value of $r$ that guarantees the correspondence between solutions of (1) and global solutions of (5), is given by the number $\hat{r}$ defined in (8). However, computational tests show that this value is usually larger than the actual value needed to obtain a solution to problem (1). Following the idea presented in [16] and [7], we use an incremental rank scheme starting with $r \ll \hat{r}$, and we employ the global optimality condition of Proposition 1 to prove optimality of the current solution.

Concerning the method for finding a stationary point of problem (14), we select a gradient type method defined by an iteration of the form

$$
v_i^{k+1} = v_i^k - \alpha^k \nabla_{v_i} f_\varepsilon(v^k) \quad i = 1, \ldots, n, \tag{22}
$$

where $\alpha^k \in (0, \alpha_M]$, for some $\alpha_M > 0$, is obtained by a suitable line-search procedure satisfying

$$f_\varepsilon(v^{k+1}) \le f_\varepsilon(v^0), \qquad (23)$$

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

The choice of a gradient type method for the minimization of $f_\epsilon$ guarantees (see Proposition 3 below) that, for $\varepsilon$ sufficiently large, the whole sequence of iterations stays in the set $\{v \in \mathbb{R}^{nr} : \|v_i\|^2 \ge 1, \ i = 1, \ldots, n\}$. Hence, for $\varepsilon$ sufficiently large, the barrier term (13) reduces to a constant, thus avoiding the annoying effect of getting close to the boundary of $S_\delta$, which might negatively affect the behavior of the algorithm.

**Proposition 3** *Let $\{v^k\}$ be the sequence generated with the iterative scheme (22) for a given $v^0 \in \mathcal{F}$, where each $\alpha^k$ satisfies (23) and $\alpha^k \le \alpha_M$. Then, there exists $\bar\varepsilon > 0$ such that, for every $\varepsilon \ge \bar\varepsilon$, we have, for all $k$,*

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

*Proof* By (23), for a fixed value $\varepsilon > 0$ the sequence $\{v^k\}$ stays in the compact level set $\mathcal{L}_\varepsilon(v^0)$. The proof is by induction. Assume that there exists $\bar\varepsilon > 0$ such that, for any $\varepsilon \ge \bar\varepsilon$, it is true that $\|v_i^k\|^2 \ge 1$. We show that the same is true also for $k$ replaced $k + 1$. We can write

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

where we use (18). If $\|v_i^k\| = 1$, then $\|v_i^{k+1}\|^2 \ge 1$. Otherwise, if $\|v_i^k\| > 1$, we need to verify that a value of $\bar\varepsilon$ exists such that, for all $\varepsilon \ge \bar\varepsilon$,

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

namely

$$1 - \frac{8\alpha_M}{\varepsilon\delta^2}\|v_i^k\|^2 \ge 0. \qquad (24)$$

By Proposition 2, we have that $\|v_i^k\|^2 \le (2\overline{C}\varepsilon\delta^2)^{\frac{1}{2}} + 1$ for all $k$, which combined with (24) implies that $\varepsilon$ has to satisfy

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

which is possible for a sufficiently large value of $\varepsilon$. $\qquad \square$

At this point we are ready to define the algorithm SpeeDP.

ALGORITHM `SpeeDP`

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

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

   S.0 Set $r = r^j$.
   S.1 Starting from $v^0 \in \mathcal{F}$, find a stationary point $\overline{v} \in \mathbb{R}^{nr}$ of problem (14) by a gradient type method satisfying (22) and (23).
   S.2 Compute the multiplier $\overline{\lambda}$ with (10).
   S.3 Determine the minimum eigenvalue $\mu_{\min}(\overline{\lambda})$ of $Q + \mathrm{Diag}(\overline{\lambda})$.
   S.4 If $\mu_{\min}(\overline{\lambda}) \geq -tol_\varepsilon$, then exit.

Return $\overline{v}, \overline{r} = r^j, \overline{\lambda}$, and $\mu_{\min}(\overline{\lambda})$.

`SpeeDP` returns $\overline{v}$ and $\mu_{\min}(\overline{\lambda})$. If $\mu_{\min}(\overline{\lambda}) \geq -tol_\varepsilon$, then the matrix $Q + \mathrm{Diag}(\overline{\lambda})$ is positive semidefinite within a tolerance $tol_\varepsilon$ so that a solution (or a good approximation) to problem (1) is obtained as $X^* = \overline{V}\,\overline{V}^T$, where $\overline{V} \in \mathbb{R}^{n \times \overline{r}}$ is such that $\mathrm{vec}(\overline{V}^T) = \overline{v}$. Recall that the value

$$z_{LB} = -e^T \overline{\lambda} + n\mu_{\min}(\overline{\lambda}),$$

provides a lower bound to the solution of problem (1) (see, e.g, [16,31]). Incidentally, note that the bound is valid for every $\overline{\lambda}$; moreover for every $\overline{v} \in \mathbb{R}^{n\overline{r}}$, if $\overline{\lambda}$ is computed by means of (10), it is easy to check that $-e^T \overline{\lambda} = Q \bullet \overline{V}\,\overline{V}^T$.

Finally, we want to stress the advantages of the `SpeeDP` algorithm based on the merit function $f_\varepsilon$ over the scheme `EXPA` based on the penalty function (7) introduced in [16].

- The theoretical properties of the exact penalty function (7) depend on the penalty parameter $\eta$ that is required to be smaller than a certain threshold value. However, choosing a small value of $\eta$ may negatively affect both the efficiency and the accuracy of the algorithm `EXPA`. On the contrary, the equivalence properties of the merit function introduced in this paper hold for any value of the parameter $\epsilon > 0$.
- Each computation of the penalty function (7) requires the evaluation of the multiplier function $\lambda(v)$ as in (10) which is not needed in the function $f_\epsilon$. This implies that the computation of $f_\epsilon$ requires less matrix-vector products than the evaluation of the function (7), with a significant reduction of computational time.

All these advantages are supported by the numerical results reported in Sect. 6.

## 5 `SpeeDP-MC`: a heuristic for large scale Max-Cut instances

In this section we combine the algorithm `SpeeDP` for computing the SDP bound with a heuristic for the Max-Cut problem (3). Note that, differently from the previous sections, here we address a *maximization* problem. Therefore, the meanings of the terms "lower" and "upper" bound have now to be referred to problem (3).

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

Let $X$ be the optimal solution to (1) and let $v_i \in \mathbb{R}^r$, $i = 1, \ldots, n$, be vectors whose Gramian matrix coincides with $X$. Recall that the Goemans–Williamson (GW) algorithm outputs the node bipartition $(S, N \backslash S)$ with $S = \{i : h^T v_i \geq 0\}$, where $h \in \mathbb{R}^r$ is generated from a uniform random distribution on the unit sphere $\{x \in \mathbb{R}^r : \|x\| = 1\}$.

We choose the GW heuristic for two main reasons. First, because it has a good theoretical approximation ratio. Indeed, the expected weight of $(S, N \backslash S)$ is shown in [12] to be at most 12.1 % below the optimal value of (1) for instances with a nonnegative weighted adjacency matrix (a more general case is treated in [27,28]). Second, because the only ingredient necessary to run the algorithm is the set of vectors $v_i \in \mathbb{R}^r$, $i = 1, \ldots, n$, which are precisely the `SpeeDP` output. Hence `SpeeDP` makes it possible to apply the GW approximation algorithm to very large graphs since, on the one hand, it is able to solve problem (1) in a reasonable amount of time also for very large graphs and, on the other hand, it provides the vectors $v_i$, $i = 1, \ldots, n$, "for free".

To the contrary, once the SDP bound has been computed with interior point methods, additional computational effort is necessary to produce the vectors $v_i$, with $i = 1, \ldots, n$.

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 moves are made until no further improvement is possible.

In [10], where a similar heuristic is described but problem (1) is solved by an 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 A graph $G = (N, E)$, its Laplacian matrix $L$, $\alpha > 0$, $k_{\max}$.
Initialization Set $Q = -\frac{1}{4}L$, $\hat{x} = e$ and $\overline{\beta} = \alpha \sum_{i,j} |q_{ij}|/|E|$.
For $k = k_{max}, \ldots, 0$ do:

S.0 Set $\beta = k\overline{\beta}$ 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 $z_{LB}$ the corresponding computed lower bound.
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 $-z_{LB}$.

Note that the amount of perturbation decreases after each iteration until it gets to zero. We stress that repeating step 1 several times is not as expensive as it may appear, because we make use of a warm start technique: beginning from the second iteration, we start `SpeeDP` from the solution found at the previous step, so that each computation of the minimum is sensibly 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.

## 6 Numerical results

In this section, we describe our computational experience both with algorithm `SpeeDP` for solving problem (1), and with the heuristic `SpeeDP-MC` for finding good Max-Cut solutions for large graphs.

`SpeeDP` is implemented in Fortran 90 and all the experiments have been run on a PC with 2 GB of RAM.

The parameters $\delta$ and $\varepsilon$ in (14) have been set to 0.25 and $10^3 \cdot \delta^{-1}$, respectively. The tolerance $tol_\varepsilon$ has been set to $10^{-3}$.

For the unconstrained optimization procedure we use a Fortran 90 implementation of the non-monotone version of the Barzilai–Borwein method proposed in [17] which satisfies (22) and (23).

As for the choice of the starting value $r^1$ of the rank, we use the same rule based on $n$ as in [16] using values $8 \leq r^1 \leq 30$ for $n$ from 100 up to more than 20000. The updating rule for the rank $r^j$ is simple $r^{j+1} = \min\{\lfloor r^j \cdot 1.5 \rfloor, \hat{r}\}$ where $\hat{r}$ is given in (8).

Positive semidefiniteness of $Q + \text{Diag}(\overline{\lambda})$ is checked by means of the subroutines `dsaupd` and `dseupd` of the `ARPACK` library.

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.[2] The code DSDP is considered particularly efficient for solving problems with low-rank structure and sparsity in the data (as it is the case for Max-Cut instances). In addition, DSDP has relatively low memory requirements for an interior point method, and is indeed able to solve instances up to around 10000 nodes. As a Spectral Bundle method, we use SB, which can be found in [19] and is downloadable from the web page.[3]

To make a comparison with other NLP based methods, we choose the code SDPLR-MC, proposed by Burer and Monteiro in [7], downloadable from the web page,[4] and the code EXPA proposed in [16].

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 (5). We remark that SDPLR-MC does not check the global optimality condition $Q + \text{Diag}(\bar{\lambda}) \succeq 0$, while both EXPA and SpeeDP do.

Our benchmark set consists of standard instances of the Max-Cut problem with number of nodes ranging from 100 to 20000 and 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.[5] The second set of problems belongs to the group Gset of randomly generated problems by means of the machine-independent graph generator *rudy* [33]. These problems can also be downloaded from Burer's web page (see footnote 4).

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 comparisons with DSDP.

We compare the different codes on the basis of the level of accuracy and of the computational time. Besides reporting detailed results in tables, we use a graphical view of the results by using the *performance profile*, proposed in [9]. Given a set of solvers $S$ and a set of problems $P$, we compare the performance of a solver $s \in S$ on problem $p \in P$ against the best performance obtained by any solver in $S$ on the same problem. To this end we define the performance ratio

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

where $t_{p,s}$ is the performance criterion used, and we consider a cumulative distribution function $\rho_s(\tau) = \frac{1}{|P|} |\{p \in P : r_{p,s} \leq \tau\}|$. Then we draw $\rho_s(\tau)$ with respect to the parameter $\tau$ that is represented on the $x$-axis with a logarithmic scale. The 'higher' the resulting curve the better is the corresponding method with respect to the criterion chosen; efficiency is measured by how fast the curve reaches the value of 1 (since all

[2] http://www-unix.mcs.anl.gov/DSDP/.

[3] http://www-user.tu-chemnitz.de/~helmberg/.

[4] http://dollar.biz.uiowa.edu/~sburer/software/SDPLR.

[5] http://euler.nmt.edu/~brian/sdplib.

**Table 1** Optimal values

| PROBLEM | SpeeDP | EXPA | SDPLR_MC | SB | DSDP | |
|---|---|---|---|---|---|---|
| | PRIMAL | PRIMAL | PRIMAL | DUAL | PRIMAL | DUAL |
| mcp100 | −226.15735 | −226.15734 | −226.15129 | −226.15923 | −226.15733 | −226.15735 |
| mcp124-1 | −141.99048 | −141.99041 | −141.99003 | −141.99370 | −141.99044 | −141.99048 |
| mcp124-2 | −269.88017 | −269.88016 | −269.88016 | −269.88222 | −269.88012 | −269.88017 |
| mcp124-3 | −467.75011 | −467.75010 | −467.75009 | −467.75370 | −467.75004 | −467.75012 |
| mcp124-4 | −864.41186 | −864.41183 | −864.41181 | −864.41997 | −864.41166 | −864.41187 |
| mcp250-1 | −317.26434 | −317.26421 | −317.26431 | −317.27079 | −317.26429 | −317.26435 |
| mcp250-2 | −531.93008 | −531.93001 | −531.92973 | −531.93491 | −531.92998 | −531.93009 |
| mcp250-3 | −981.17257 | −981.17248 | −981.17239 | −981.17796 | −981.17239 | −981.17257 |
| mcp250-4 | −1681.9601 | −1681.9597 | −1681.9570 | −1681.9750 | −1681.9600 | −1681.9601 |
| mcp500-1 | −598.14849 | −598.14798 | −598.14800 | −598.15877 | −598.14840 | −598.14852 |
| mcp500-2 | −1070.0566 | −1070.0562 | −1045.0727 | −1070.0759 | −1070.0566 | −1070.0568 |
| mcp500-3 | −1847.9700 | −1847.9694 | −1847.9666 | −1847.9836 | −1847.9695 | −1847.9700 |
| mcp500-4 | −3566.7376 | −3566.7357 | −3566.7334 | −3566.7479 | −3566.7377 | −3566.7381 |
| G01_mc | −12083.198 | −12083.197 | −12083.042 | −12083.265 | −12083.196 | −12083.198 |
| G11_mc | −629.16298 | −629.14611 | −629.15995 | − 629.17007 | −629.16472 | −629.16478 |
| G14_mc | −3191.5667 | −3191.5654 | −3191.5633 | −3191.5847 | −3191.5661 | −3191.5668 |
| G22_mc | −14135.946 | −14135.943 | −14135.867 | −14136.044 | −14135.945 | −14135.946 |
| G32_mc | −1567.6303 | −1567.5895 | −1567.6323 | −1567.6519 | −1567.6394 | −1567.6397 |
| G35_mc | −8014.7388 | −8014.7379 | −8014.7307 | −8014.8070 | −8014.7376 | −8014.7397 |
| G36_mc | −8005.9552 | −8005.9512 | −8005.9483 | −8006.0213 | −8005.9632 | −8005.9638 |
| G43_mc | −7032.2217 | −7032.2196 | −7032.2078 | −7032.2749 | −7032.2208 | −7032.2219 |
| G48_mc | −5999.9993 | −5999.9968 | −5999.9662 | −6000.0000 | −5999.9985 | −6000.0000 |
| G51_mc | −4006.2553 | −4006.2533 | −4006.2537 | −4006.2745 | −4006.2546 | −4006.2555 |
| G52_mc | −4009.6384 | −4009.6380 | −4009.6202 | −4009.6574 | −4009.6383 | −4009.6388 |
| G55_mc | −11039.460 | −11039.450 | −11039.341 | −11040.159 | −11039.449 | −11039.461 |
| G57_mc | −3885.4783 | −3885.3318 | −3885.4501 | −3885.5189 | −3885.4868 | −3885.4892 |
| G58_mc | −20135.875 | −20135.854 | −20136.032 | −20136.287 | −20136.181 | −20136.190 |
| G60_mc | −15222.239 | −15222.220 | −15222.138 | −15223.193 | −15222.257 | −15222.268 |
| G62_mc | −5430.8903 | −5430.6629 | −5430.8413 | −5430.9512 | −5430.9084 | −5430.9104 |
| G63_mc | −28243.308 | −28243.218 | −28243.876 | −28244.577 | −28244.406 | −28244.418 |
| G64_mc | −10465.836 | −10465.804 | −10465.868 | −10465.970 | −10465.898 | −10465.904 |
| G65_mc | −6205.4852 | −6205.2216 | −6205.4434 | −6205.5822 | −6205.5322 | −6205.5382 |
| G66_mc | −7077.1819 | −7077.0132 | −7077.1139 | −7077.2640 | −7077.2090 | −7077.2137 |
| G67_mc | −7744.3288 | −7744.2624 | −7744.3011 | −7744.4942 | −7744.4245 | −7744.4365 |
| G70_mc | −9861.5209 | −9861.4825 | −9861.3992 | −9861.7340 | −9861.5143 | −9861.5246 |
| G72_mc | −7808.3993 | −7808.1436 | −7808.4139 | −7808.5914 | −7808.5343 | −7808.5393 |
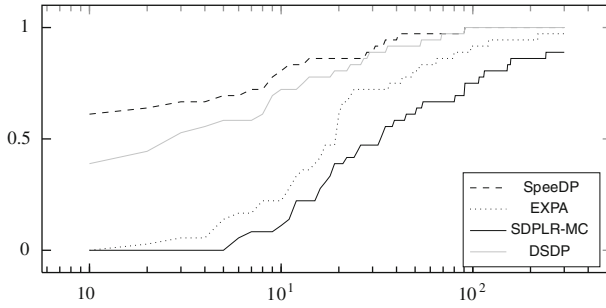
**Fig. 1** Performance profile with performance criterion equal to primal–dual gap
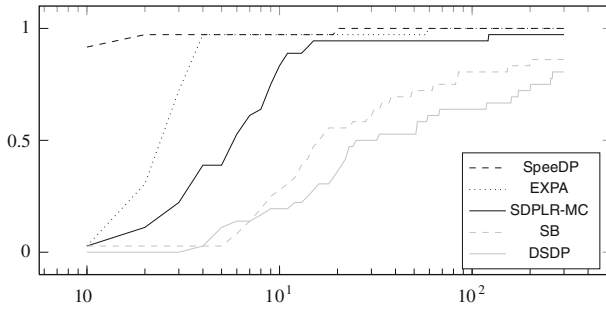


**Fig. 2** CPU time performance profiles

the methods solve all the problems, all the methods eventually reach the performance value 1 if a sufficiently large $\tau$ is allowed).

As for the accuracy, following [25], we report the primal and/or the dual objective function values obtained by the five methods in Table 1. DSDP gives both primal and dual values in output, SpeeDP, EXPA, and SDPLR-MC return the primal objective value only, while SB produces a value of the dual objective function that is a bound on the optimal value of problem (1). Furthermore, we plot the performance profile in Fig. 1, choosing the relative duality gap as a performance criterion. In particular, we consider the relative difference between the primal value $f_{p,s}^*$ of any solver $s$ on problem $p$ and dual value $f_{p,\text{DSDP}}^*$ given by DSDP, namely we set

$$t_{p,s} = \frac{f_{p,s}^* - f_{p,\text{DSDP}}^*}{1 + |f_{p,s}^*| + |f_{p,\text{DSDP}}^*|}.$$

By analyzing them, it emerges that, as for the accuracy, SpeeDP can be considered comparable with DSDP, whereas EXPA, SDPLR-MC, and SB are usually worse.

As for the computational efficiency, we do not report the CPU times of our computational experiments explicitly; instead in Fig. 2 we give an overall view of the results by showing the performance profile of the five methods where the performance criterion $t_{p,s}$ is the CPU time in seconds needed by solver $s$ to solve problem $p$.

**Table 2** Optimal values and computation time for G77 and G81

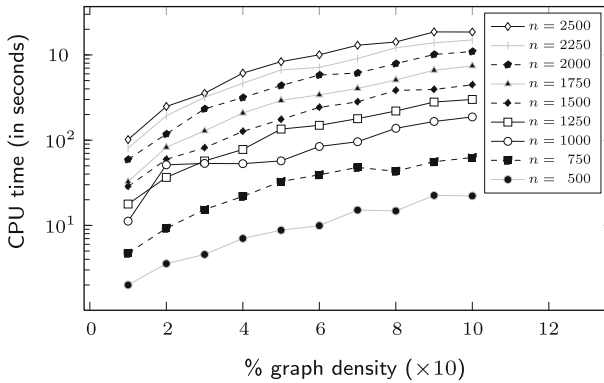| PROB | ACCURACY | | | | TIME | | | |
|------|-------------------|------------------|---------------------|---------|---------|------|----------|---------|
|      | PRIMAL<br>SpeeDP | PRIMAL<br>EXPA | PRIMAL<br>SDPLR_MC | DUAL SB | SpeeDP | EXPA | SDPLR_MC | SB |
| G77 | $-11045.6$ | $-11045.1$ | $-11045.4$ | $-11045.8$ | 57.8 | 54.2 | 71.7 | 4260.6 |
| G81 | $-15656.1$ | $-15655.6$ | $-15655.8$ | $-15656.3$ | 64.7 | 86.5 | 108.3 | 33538.2 |



**Fig. 3** Average `SpeeDP-MC` CPU time for random graphs

The results related to the two problems G77 and G81 are reported separately in Table 2. It emerges from these profiles and from the table that `SpeeDP` outperforms all the other methods.

Finally, we report on the numerical results obtained by the heuristic algorithm `SpeeDP-MC` described in Sect. 5 applied to some large random graphs. `SpeeDP-MC` is implemented in C and uses the Fortran 90 version of `SpeeDP` as a routine. As for the parameters, in the experiments we set $k_{\max} = 2$ and $\alpha = 10^{-2}$. We generated 500 random vectors $h$ in the Goemans–Williamson heuristic and we selected the one that provides the best cut. We used the graph generator *rudy* [33] to produce instances with several sizes and densities and different weights. We first considered graphs with node sizes $n$ equal to $500 + i \cdot 250$, for $i = 0, \ldots, 8$ and with edge *densities* 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. Details on the results can be found in [14]. We draw in Fig. 3 the average CPU time and in Fig. 4 the gap obtained, as a function of the graph density.

As it emerges from the figures, the heuristic is able to produce a good cut in a small amount of time. As expected, the performance of the heuristic is better on sparse graphs in term of time, but the gap decreases when the graph density increases. The average gap on all instances is 1.5 % and the average CPU time is 340.6 s, while the average gap obtained by a plain Goemans–Williamson algorithm (no 1-opt improvement and $k_{\max} = 0$) is 2.04 % with an average CPU time of 95.7 s.
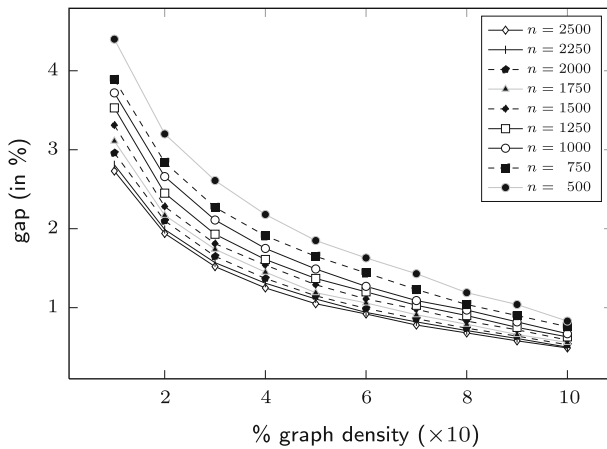
**Fig. 4** Average `SpeeDP-MC` gap for random graphs

**Table 3** Random sparse graphs with 100001 nodes and 7050827 edges

| WEIGHTS | UPPER BOUND | BEST CUT | SPEEDP-MC | | PLAIN GW | |
|---|---|---|---|---|---|---|
| | | | CPU TIME | GAP% | CPU TIME | GAP% |
| 1 | 4110550.6 | 3962681 | 16961 | 3.60 | 1182 | 5.24 |
| [1, 100] | 212084176.3 | 203457740 | 18202 | 4.07 | 1208 | 5.97 |
| [−1000, 1000] | 21006828446.4 | 20146336217 | 17055 | 4.10 | 1183 | 5.99 |

Furthermore, we consider huge graphs, in order to explore how far we can go with the number of nodes. For this set of instances, we run `SpeeDP-MC` on a machine with 6 GB of RAM and we set $k_{max} = 5$ and $\alpha = 10^{-2}$.

We generate three random graphs with 100001 nodes, 7050827 edges and different edge weights. For this set of graphs, given the huge size, we choose not to increment $r$ in `SpeeDP` and we set $r = 60$. The results are shown in Table 3, where we report the weight ranges, the upper bound value, the weight of the best cut obtained by `SpeeDP`-MC, and finally the total CPU time and the % gap for both `SpeeDP`-MC and the plain GW heuristic.

We also generated some 6-regular graphs (3D toroidal grid graphs) with 1030301 nodes, 3090903 edges, and different edge weights. Also in this case $r$ was not allowed to change and was set to a value of 90. The results are reported in Table 4. To the best of our knowledge, no other methods can achieve this accuracy for graphs of comparable size.

## 7 Concluding remarks and future work

In this paper we define `SpeeDP`, a fast globally convergent algorithm for solving problem (1), which belongs to the family of low rank nonlinear programming approaches.

**Table 4** 6-regular graphs with 1030301 nodes and 3090903 edges

| WEIGHTS | UPPER BOUND | BEST CUT | SpeeDP- MC | | PLAIN GW | |
|---|---|---|---|---|---|---|
| | | | CPU TIME | GAP% | CPU TIME | GAP% |
| 1 | 3090151.2 | 3060300 | 2710 | 0.97 | 1401 | 0.98 |
| [1, 10] | 15454781.5 | 15337609 | 5391 | 0.76 | 1445 | 0.88 |
| [1, 1000] | 1545558266.8 | 1534308113 | 8943 | 0.73 | 1636 | 0.85 |
| [−100, 100] | 57317884.9 | 49012182 | 13683 | 14.49 | 1648 | 26.31 |

`SpeeDP` outperforms existing methods for solving the special structured semidefinite programming problem (1) and provides both a primal and an approximate dual bound. We also define a heuristic algorithm for Max-Cut, which is a straightforward enhancement of the Goemans–Williamson method, and is able to handle graphs with up to millions of nodes and edges. The heuristic produces both a feasible cut and a valid bound, hence it can certify the maximal deviation of the weight of this cut from the optimum. As a future step, we plan to include `SpeeDP` in a branch-and-bound scheme similarly to what has been done for an interior point method in the `BiqMac` code of [32]. This way, we aim at increasing the size of the Max-Cut instances that can be solved exactly by semidefinite programming.

# References

1. Barvinok, A.: Problems of distance geometry and convex properties of quadratic maps. Discret. Comput. Geom. **13**, 189–202 (1995)
2. Benson, H.Y., Vanderbei, R.J.: On formulating semidefinite programming problems as smooth convex nonlinear optimization problems. Technical Report ORFE 1999–01, Princeton University, NJ (1999)
3. Benson, S.J., Ye, Y., Zhang, X.: Solving large-scale sparse semidefinite programs for combinatorial optimization. SIAM J. Optim. **10**(2), 443–461 (2000)
4. Bertsekas, D.P.: Nonlinear Programming. Athena Scientific, Belmont (1999)
5. Burer, S., Monteiro, R.D., Zhang, Y.: Solving a class of semidefinite programs via nonlinear programming. Math. Program. **93**, 97–122 (2002)
6. Burer, S., Monteiro, R.D.C.: A projected gradient algorithm for solving the maxcut sdp relaxation. Optim. Methods Softw. **15**(3–4), 175–200 (2001)
7. Burer, S., Monteiro, R.D.C.: A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. Math. Program. **95**, 329–357 (2003)
8. Delorme, C., Poljak, S.: Laplacian eigenvalues and the maximum cut problem. Math. Program. **62**(3), 557–574 (1993)
9. Dolan, E., Morè, J.: Benchmarking optimization software with performance profile. Math. Program. **91**, 201–213 (2002)
10. Fischer, I., Gruber, G., Rendl, F., Sotirov, R.: Computational experience with a bundle approach for semidefinite cutting plane relaxations of Max-Cut and equipartition. Math. Program. **105**(2–3), 451–469 (2006)
11. Fletcher, R.: Semi-definite matrix constraints in optimization. SIAM J. Cont. Optim. **23**, 493–513 (1985)
12. Goemans, M.X., Williamson, D.P.: Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. J. ACM **42**(6), 1115–1145 (1995)

13. Grippo, L., Palagi, L., Piacentini, M., Piccialli, V.: An unconstrained approach for solving low rank SDP relaxations of {−1, 1} quadratic problems. Technical Report 1.13, Dip. di Informatica e Sistemistica A. Ruberti, Sapienza Università di Roma (2009)
14. Grippo, L., Palagi, L., Piacentini, M., Piccialli, V., Rinaldi, G.: SpeeDP: a new algorithm to compute the SDP relaxations of Max-Cut for very large graphs. Technical Report DII-UTOVRM Technical Report 13.10, University of Rome Tor Vergata (2010)
15. Grippo, L., Palagi, L., Piccialli, V.: Necessary and sufficient global optimality conditions for NLP reformulations of linear SDP problems. J. Glob. Optim. **44**(3), 339–348 (2009)
16. Grippo, L., Palagi, L., Piccialli, V.: An unconstrained minimization method for solving low-rank SDP relaxations of the maxcut problem. Math. Program. **126**, 119–146 (2011)
17. Grippo, L., Sciandrone, M.: Nonmonotone globalization techniques for the Barzilai–Borwein gradient method. Comput. Optim. Appl. **23**, 143–169 (2002)
18. Grone, R., Pierce, S., Watkins, W.: Extremal correlation matrices. Linear Algebra Appl. **134**, 63–70 (1990)
19. Helmberg, C., Rendl, F.: A spectral bundle method for semidefinite programming. SIAM J. Optim. **10**, 673–696 (2000)
20. Homer, S., Peinado, M.: Design and performance of parallel and distributed approximation algorithm for the Maxcut. J. Parallel Distrib. Comput. **46**, 48–61 (1997)
21. Journée, M., Bach, F., Absil, P., Sepulchre, R.: Low-rank optimization for semidefinite convex problems. SIAM J. Optim. **20**(5), 2327–2351 (2010)
22. Kim, S., Kojima, M.: Second order cone programming relaxation of nonconvex quadratic optimization problems. Optim. Methods Softw. **15**, 201–224 (2000)
23. Laurent, M., Poljak, S., Rendl, F.: Connections between semidefinite relaxations of the Max-Cut and stable set problems. Math. Program. **77**, 225–246 (1997)
24. Liers, F., Jünger, M., Reinelt, G., Rinaldi, G.: Computing exact ground states of hard Ising spin glass problems by branch-and-cut. In: Hartmann, A., Rieger, H. (eds.) New Optimization Algorithms in Physics, pp. 47–69. Wiley, London (2004)
25. Mittelmann, H.: An independent benchmarking of SDP and SOCP solvers. Math. Program. **95**, 407–430 (2003)
26. Muramatsu, M., Suzuki, T.: A new second-order cone programming relaxation for max-cut problems. J. Oper. Res. Jpn **46**, 2003 (2001)
27. Nesterov, Y.: Quality of semidefinite relaxation for nonconvex quadratic optimization. CORE Discussion Papers 1997019, Université Catholique de Louvain, Center for Operations Research and Econometrics (CORE) (1997)
28. Nesterov, Y.: Semidefinite relaxation and nonconvex quadratic optimization. Optim. Methods Softw. **9**(1–3), 141–160 (1998)
29. Pataki, G.: On the rank of extreme matrices in semidefinite programs and the multiplicity of optimal eigenvalues. Math. Oper. Res. **23**, 339–358 (1998)
30. Poljak, S., Rendl, F.: Solving the Max-Cut problem using eigenvalues. Discret. Appl. Math. **62**(1–3), 249–278 (1995)
31. Poljak, S., Rendl, F., Wolkowicz, H.: A recipe for semidefinite relaxation for 0-1 quadratic programming. J. Glob. Optim. **7**, 51–73 (1995)
32. Rendl, F., Rinaldi, G., Wiegele, A.: Solving Max-Cut to optimality by intersecting semidefinite and polyhedral relaxations. Math. Program. **121**(2), 307–335 (2010)
33. Rinaldi, G.: Rudy: A graph generator. http://www-user.tu-chemnitz.de/~helmberg/sdp_software.html (1998)