Computational Approaches to Max-Cut

Laura Palagi[1], Veronica Piccialli[2], Franz Rendl[3], Giovanni Rinaldi[4], and Angelika Wiegele[3]

[1] Università di Roma la Sapienza, Dipartimento di Informatica e Sistemistica A. Ruberti - via Ariosto 25 - 00185 Rome, Italy. laura.palagi@uniroma1.it
[2] Università degli Studi di Roma Tor Vergata, Dipartimento di Informatica, Sistemi e Produzione - via del Politecnico 1 - 00133 Rome, Italy. piccialli@disp.uniroma2.it
[3] Alpen-Adria-Universität Klagenfurt - Universtitätsstr. 65–67 - 9020 Klagenfurt, Austria. franz.rendl,angelika.wiegele@uni-klu.ac.at
[4] Istituto di Analisi dei Sistemi ed Informatica "A. Ruberti", CNR - viale Manzoni 30 - 00185 Rome, Italy. rinaldi@iasi.cnr.it

Summary. This chapter focuses on the impact that the advances in SDP algorithms have on the computation of good/optimal solutions of the Max-Cut problem.

Being Max-Cut NP-hard, the computation of good upper and lower bounds is the kernel of any algorithm for computing good/optimal solutions. SDP relaxations have shown to be particularly successful in the computation of both these bounds. In particular, the Goemans-Williamson algorithm of 1995, that quite substantially contributed to widen the interest for SDP based techniques, plays a fundamental role not only for Max-Cut but also for some related problems. Some of them are shortly described here like, in particular, Max-k-Cut, Coloring and Ordering problems.

The main methods available for solving the basic SDP problem are surveyed such as the interior-point methods, the spectral bundle method and, with some emphasis, a recent class of effective non-linear programming based methods.

A computational section concludes the chapter, with a comparison of the performances of the main methods described in the previous sections.

1 Introduction

Max-Cut is one of the most studied combinatorial optimization problems because of its wide range of applications and because of its connections with other fields of discrete mathematics (see, e.g., the book by Deza and Laurent [10]). Like other interesting combinatorial optimization problems, Max-Cut is very simple to state.

An undirected graph $G = (V, E)$ is given with a weight edge function $w : E \to \mathbb{R}$ encoded in its weighted adjacency matrix A. The problem is to find a cut, i.e., an edge subset

$$\delta(S) = \{e = uv \in E : u \in S, v \notin S\},$$

where S is a (possibly empty) subset of V, such that its total weight $w(\delta(S)) = \sum_{uv \in \delta(S)} a_{uv}$ is maximized. Put it differently, the problem asks for a bipartition $(S, V \setminus S)$ of the vertices V of G, so that the total weight of the edges that go across the sets of the bipartition is maximal.

The problem can be restated as an unconstrained problem on binary variables with arbitrary quadratic objective function. Despite its simple formulation, this is an NP-hard problem, therefore some pseudo-enumeration technique is essentially unavoidable if one wants to solve it to optimality. To limit the growth of the number of subproblems that such a technique typically generates, good upper and lower bounds on the optimal solution value are necessary. Upper bounds require the definition and the solution of a suitable relaxation to the problem, while lower bounds are computable by finding a 'good' cut with a heuristic algorithm.

The first computational experiments on non-trivial Max-Cut instances were carried out with Linear Programming (LP) based relaxations. More recently, as soon as the field of Semidefinite Programming (SDP) started to develop, SDP relaxations to the problem were used. The successful results obtained with SDP techniques in the efficient generation of both upper and lower bounds rapidly made Max-Cut the application of choice of SDP algorithms and greatly increased the interest on this problem among the members of the SDP community. This is the main motivation for the topic of this chapter in which some recent methods are surveyed with some emphasis on their impact on the computation of heuristic and exact solutions. To be consistent with the topic of this volume, the chapter mostly covers aspects of the SDP relaxation of Max-Cut, while the basic LP based relaxations are only briefly outlined.

The following simple semidefinite program plays a fundamental role in combinatorial optimization. For a given matrix C in the space of the $n \times n$ symmetric matrices $\mathcal{S}^n$, find $X \in \mathcal{S}^n$ such that

$$z_P = \max\{\langle C, X\rangle : \mathrm{diag}(X) = e, X \succeq 0\}, \tag{1}$$

where e is the vector of all ones, $\mathrm{diag}(X)$ is the vector containing the diagonal of X and $X \succeq 0$ indicates that X is positive semidefinite.

The associated dual problem is given as follows:

$$z_D = \min\{e^T y : \mathrm{Diag}(y) - C \succeq 0\}, \tag{2}$$

where $\mathrm{Diag}(y)$ is the diagonal matrix having as a diagonal the vector $y \in \mathbb{R}^n$. Since both the primal and the dual problem have strictly feasible points ($X = I$, the identity matrix, for the primal and $y = (\lambda_{\max}(C) + 1)e$ for the dual, where $\lambda_{\max}(C)$ denotes the largest eigenvalue of C), both the primal and the dual optimal values are attained and are equal to each other. Problem (1) is perhaps best known in connection with Max-Cut. To formulate the problem, we define the Laplacian $L = \mathrm{Diag}(Ae) - A$ that is associated with the weighted adjacency matrix A of G. Then Max-Cut can then be formulated as follows:

$$z_{MC} = \max\left\{\frac{1}{4}x^T Lx : x \in \{-1,1\}^n\right\}. \tag{3}$$

Bipartitions $(S, V \setminus S)$ are encoded by n-dimensional vectors x with $x_i = 1$ for $i \in S$ and $x_i = -1$ for $i \notin S$. It is a well-known fact and easy to verify that in this case

$$\sum_{uv\in\delta(S)} a_{uv} = \frac{1}{4}x^T Lx.$$

Since $x^T Lx = \langle L, xx^T\rangle$ and $xx^T \succeq 0$ with $\mathrm{diag}(xx^T) = e$, it is clear that problem (1) with $C = \frac{1}{4}L$ provides a relaxation of Max-Cut.

The results presented on Max-Cut, and those on the related problems described in Section 2, all have some connection to the simple semidefinite program (1).

In Section 3 the main methods are described that are used to solve (1). In particular, Subsection 3.1 is devoted to the classic solution method, the one based on the interior-point approach. Aiming at finding methods that are better suited for problems with a sparse objective function matrix, some reformulations of (1) and of its dual (2) have been recently studied that avoid semidefinite matrices and turn out to be quite useful from a computational point of view.

A first natural way to eliminate the semidefiniteness constraint $\mathrm{Diag}(y) - C \succeq 0$ in the dual problem is as follows. We first observe that the primal constraints $\mathrm{diag}(X) = e$ imply $\mathrm{tr}(X) = n$, where $\mathrm{tr}(X)$ is the trace of matrix X. Adding this redundant constraints to (1) leads to the following dual problem, with additional dual variable λ for the redundant constraint:

$$\min\{e^T y + n\lambda : \lambda I + \mathrm{Diag}(y) - C \succeq 0\}.$$

For any feasible solution of this problem we have

$$\lambda_{\max}(C - \mathrm{Diag}(y) - \lambda I) \leq 0,$$

hence

$$\lambda \geq \lambda_{\max}(C - \mathrm{Diag}(y)).$$

Therefore, minimizing $e^T y + n\lambda$ forces equality and yields the following equivalent formulation of the dual:

$$\min_{y\in\mathbb{R}^n} e^T y + n\lambda_{\max}(C - \mathrm{Diag}(y)).$$

We can slightly simplify this problem by removing the linear term. We represent $y \in \mathbb{R}^n$ through its components parallel and orthogonal to e,

$$y = \alpha e + v \text{ where } \langle v, e\rangle = 0 \text{ and } \alpha \in \mathbb{R}. \tag{4}$$

In this case the dual simplifies to

$$z_D = \min\{n\lambda_{\max}(C - \mathrm{Diag}(v)) : \langle v, e\rangle = 0\}. \tag{5}$$

This is a convex but non-smooth minimization problem, which can be solved with subgradient techniques from convex optimization. In Subsection 3.2 we briefly discuss the spectral bundle method, which is tailored to solve problems of this type.

A second reformulation is based on the observation that $X \succeq 0$ is equivalent to $X = VV^T$ for some $n \times r$ matrix V. We denote the columns of V^T by v_i for $i = 1, \ldots, n$, i.e., $V^T = (v_1, \ldots, v_n)$ and $v_i \in \mathbb{R}^r$. In this case, $\mathrm{diag}(VV^T) = e$ is equivalent to $\|v_i\|^2 = 1$ for $i = 1, \ldots, n$. This gives the non-linear problem

$$\max \left\{ \sum_{ij} c_{ij} \langle v_i, v_j \rangle : v_i \in \mathbb{R}^r, \|v_i\|^2 = 1,\ i = 1, \ldots, n \right\}. \tag{6}$$

This formulation (in minimization form) is investigated in Subsection 3.3, and is the basis for the famous Goemans-Williamson hyperplane rounding technique that will be described in Section 4 (see also the chapter by Chlamtac and Tulsiani in this handbook). Because of their computational efficiency most of Section 3 is devoted to the methods based on this reformulation.

The methods described in Section 3 are used in Section 4 to find approximate solutions to Max-Cut and in Section 5 to solve the problem to optimality. Finally, some computational results for the algorithms described in these two sections are reported in Section 6. The results of the last few years obtained by exploiting the reformulation (6) are perhaps the most interesting achievements, as far as the computation of an optimal solution to (3) is concerned. For this reason, a large part of this section is devoted to them.

2 Extensions

In this section we will describe several types of problems which all have semidefinite relaxations containing the basic problem (1) in addition to other constraints.

2.1 Max-k-Cut and Coloring

Max-k-Cut and Coloring can be viewed as partition problems in graphs. Max-k-Cut takes the weighted adjacency matrix A of a weighted graph G as input and asks to partition the vertices V of G into k sets $(S_1, \ldots, S_k)$ so as to maximize the weight of all edges joining distinct partition blocks. Coloring, or more specifically k-Coloring, asks to partition V into (at most) k stable sets of G.

It seems natural to represent the partition $(S_1, \ldots, S_k)$ of V by the incidence vectors x_i of S_i which are collected in the partition matrix $X =$

$(x_1, \ldots, x_k)$. Since each vertex in V is in exactly one of the sets, the row sums of this matrix give the all-ones vector e. Hence k-partitions of V are in one-to-one correspondence with the set

$$\mathcal{P}_k = \{X : X = (x_{ij})_{n\times k}, x_{ij} \in \{0,1\}, Xe = e\}.$$

If the cardinalities of the partition blocks are also specified, we get additional (linear) constraints on the column sums of X.

Furthermore, for a k-partition matrix X we have $\mathrm{diag}(XX^T) = e$ and $kXX^T - J \succeq 0$, where $J = ee^T$.

The block-diagonal structure of XX^T also shows that $\langle A, XX^T\rangle$ is twice the weight of edges inside partition blocks. Using the Laplacian L, associated to A, it follows that

$$\frac{1}{2}\langle L, XX^T\rangle$$

is the weight of edges joining distinct partition blocks. This leads to the following semidefinite relaxation for Max-k-Cut after the variable transformation $Y = XX^T$:

$$\max\left\{\frac{1}{2}\langle L, Y\rangle : \mathrm{diag}(Y) = e, Y - \frac{1}{k}J \succeq 0, Y \geq 0\right\}.$$

Finally, the transformation $Z = \frac{k}{k-1}(Y - \frac{1}{k}J)$ leads to

$$\max\left\{\frac{k}{2(k-1)}\langle L, Z\rangle : \mathrm{diag}(Z) = e, Z \succeq 0, z_{ij} \geq -\frac{1}{k-1} \text{ for } i < j\right\}.$$

This formulation contains again the constraints $\mathrm{diag}(Z) = e, Z \succeq 0$ from the basic relaxation (1). Frieze and Jerrum [13] adopted the Goemans-Williamson hyperplane rounding idea for this relaxation and provided performance guarantees α_k, which were slightly improved later by De Klerk, Pasechnik and Warners [9]. Here are these ratios for small k: $\alpha_5 \approx 0.87661$, $\alpha_4 \approx 0.85749$, $\alpha_3 \approx 0.83601$ and $\alpha_2 = 0.87856$, which corresponds to the Goemans-Williamson ratio for Max-Cut. From a practical point of view, we refer to Ghaddar, Anjos and Liers [14] who provide computational experience for the Max-k-Cut problem using relaxations based on semidefinite optimization in combination with branch and cut.

Turning to k-Coloring, we again look for a k-partition $(S_1, \ldots, S_k)$, but in this case each S_i also has to be a stable set. Thus, if x_i represents the stable set S_i, we need to impose the constraint

$$(x_i)_u(x_i)_v = 0 \text{ for all } uv \in E(G),$$

to insure that for the edges uv, we cannot have both endpoints in S_i.

The smallest k such that the graph G has a k-Coloring is usually called the *chromatic number* $\chi(G)$ of G. We therefore have

$$\chi(G) \leq \min\left\{k : Y - \frac{1}{k}J \succeq 0, \mathrm{diag}(Y) = e, y_{uv} = 0 \text{ for all } uv \in E(G)\right\}.$$

Using the above transformation again, we get

$$\min\{\alpha : \mathrm{diag}(Z) = e, Z \succeq 0, z_{uv} = \alpha \text{ for all } uv \in E(G)\}.$$

We point out once more that this relaxation has the basic structure of (1). Karger, Motwani and Sudan [30] use this model to adopt the hyperplane rounding idea to get approximation results for Coloring. They also show that this model corresponds, in disguised form, to the Lovász Theta number, see [34].

2.2 Ordering Problem

Ordering problems assign a profit to each ordering (of n objects) and ask for an ordering of maximum profit. In the simplest case, the profit of an ordering is simply built up by a profit vector $c = (c_{ij}), i \neq j$. In this case the profit is given by $\sum_{i \prec j} c_{ij}$ where the summation is over all terms i before j in the ordering. This type of problem is called *linear ordering problem*. (See also the chapter by Anjos and Liers in this volume.)

Since we will consider matrix relaxations of this problem, it is natural to consider cost functions that depend quadratically on the ordering. In other words, we allow the profit of an ordering to be built up by terms $c_{ij,rs}$, provided i is before j, and r is before s in the ordering. We call this a *quadratic ordering problem*. Since we may encode orderings by permutations, this problem asks for a permutation of the elements of the set $S = \{1, \ldots, n\}$ such that some quadratic objective function is maximized.

In order to formulate the problem we introduce the following notation. Let $\mathcal{P}_n$ be the set of all permutations of the elements of S. For a permutation $\pi \in \mathcal{P}_n$, we define a characteristic vector $\chi(\pi) \in \{0,1\}^{\binom{n}{2}}$ of the underlying ordering as follows: for all $1 \leq i < j \leq n$, we set

$$\chi(\pi)_{ij} = \begin{cases} 1 & \text{if } \pi(i) < \pi(j) \\ 0 & \text{otherwise.} \end{cases}$$

Let $\mathrm{LO}(n)$ denote the (linear) ordering polytope on n elements, i.e., the polytope

$$\mathrm{LO}(n) = \mathrm{conv}\{\chi(\pi) \mid \pi \in \mathcal{P}_n\} \subseteq \mathbb{R}^{\binom{n}{2}}.$$

The quadratic ordering problem can then be stated as

$$\max\left\{x^\top C x : x \in \mathrm{LO}(n) \cap \{0,1\}^{\binom{n}{2}}\right\}, \tag{7}$$

where C is a real $\binom{n}{2} \times \binom{n}{2}$ matrix. For orderings clearly transitivity must hold, i.e., if i is before j and j is before k then i must be before k. To model

transitivity, it is sufficient to forbid directed 3-cycles. This is achieved by asking that the 3-dicycle inequalities

$$0 \leq x_{ij} + x_{jk} - x_{ik} \leq 1 \tag{8}$$

hold for all $i < j < k$ [22, 38].

When moving to matrix relaxations, it turns out to be more convenient to model the characteristic vectors of orderings through variables taking the values -1 and 1. The variable transformation $y_{ij} = 2x_{ij} - 1$ leads to the following form of the 3-dicycle inequalities

$$-1 \leq y_{ij} + y_{jk} - y_{ik} \leq 1.$$

Since the y_{ij}'s are -1 or 1, we have in fact that

$$|y_{ij} + y_{ik} - y_{jk}| = 1.$$

This thus shows that inequalities (8) are satisfied for $x \in \{0,1\}^{\binom{n}{2}}$ if and only if the equations

$$y_{ij}y_{jk} - y_{ij}y_{ik} - y_{ik}y_{jk} = -1 \tag{9}$$

hold for all $i < j < k$, and $y_{ij} = 2x_{ij} - 1$. In [7] it is shown that the quadratic ordering problem can be formulated as a Max-Cut problem of a graph on $\binom{n}{2} + 1$ nodes with additional constraints (9). In particular, it is shown that these constraints induce a face of the cut polytope. Thus, a relaxation of the quadratic ordering problem in $\{-1, 1\}$ variables y_{ij} is obtained by weakening the condition $Y = yy^T$ to $Y - yy^T \succeq 0$. This last condition is equivalent to $Z = \begin{pmatrix} 1 & y^T \\ y & Y \end{pmatrix} \succeq 0$. Hence we get

$$\max\left\{\langle Q, Z\rangle : \operatorname{diag}(Z) = e, -z_{ij,ik} - z_{ik,jk} + z_{ij,jk} = -1, Z \succeq 0\right\},$$

with suitably chosen cost matrix Q. Note that matrix Z is of dimension $\binom{n}{2}+1$, i.e., it is a matrix indexed by all pairs (i, j), $i < j$, plus an additional row and column. This again is the semidefinite program (1) with some additional linear constraints.

Clearly, algorithms for solving the Max-Cut problem can be used for solving the quadratic ordering problem. An extended version of the algorithm BiqMac (see Section 5) has been used to solve special instances of this problem arising from bipartite crossing minimization. This extended version is capable of treating several additional constraints, in our case equalities (9). In [7] it is demonstrated that using this semidefinite programming based algorithm clearly outperforms other existing approaches, like standard linearization or branch-and-cut. We also refer to [28] for recent strengthenings and extensions of this approach.

3 How to Solve the Semidefinite Programs

Semidefinite optimization problems have their nonlinearity only in the constraint that the primal matrix X should be contained in the closed, convex cone of semidefinite matrices. Newton method is the basis for a family of interior-point based algorithms to solve SDP. We briefly recall their key features in Subsection 3.1 below. The reformulation of the dual problem as eigenvalue optimization problem (5) is the starting point for bundle, and more specifically, spectral bundle methods. These will be briefly described in Subsection 3.2 below. Finally, the primal non-linear model (6) is the starting point for the non-linear programming based algorithms described in Subsection 3.3 below.

3.1 Interior-Point Methods

The key idea to solve semidefinite optimization problems by interior-point methods consists in applying the Newton method to the optimality conditions of the primal-dual pair of problems. We consider the following standard form of the primal-dual pair:

$$\min\{\langle C, X\rangle : \mathcal{A}(X) = b, X \succeq 0\} = \max\{b^T y : Z = C - \mathcal{A}^T(y) \succeq 0\},$$

where the linear operator $\mathcal{A}(\cdot)$ maps (symmetric) matrices to $\mathbb{R}^m$, with $(\mathcal{A}(X))_i = \langle \mathcal{A}_i, X\rangle$, and its adjoint is $\mathcal{A}^T y = \sum_i \mathcal{A}_i y_i$.

The dimension of the matrix space is n, and $b \in \mathbb{R}^m$. We assume that both problems have strictly feasible points, ensuring that both the primal and the dual optimal values are attained and are equal to each other. Interior-point methods approximately solve the parametrized optimality conditions

$$\mathcal{A}(X) = b, \quad Z + \mathcal{A}^T(y) = C, \quad ZX = \mu I, \quad \mu > 0$$

for fixed $\mu > 0$, and iterate for decreasing values of μ, until $\mu \approx 0$. The main computational effort in each iteration consists in setting up and solving a linear system for the update Δy of the dual variables. This involves several operations of order $O(n^3)$, such as computing Z^{-1} explicitly and performing several matrix multiplications of generally dense matrices of order n to determine the updates ΔX and ΔZ. Having theses updates, a line-search is carried out to arrive at a new iterate of positive definite matrices.

The linear system to be solved in case of (1) is of order n and has the simple coefficient matrix $X \circ Z^{-1}$.

A representative table that provides a flavor of the computational effort of a dual-scaling based interior-point method particularly suitable for Max-Cut is reported in Section 6.

For a complete review of recent advances in interior-point methods we refer the reader to the chapter by Alexander Engau and the chapter by Hans Mittelmann in this volume.

3.2 Spectral Bundle Method

The formulation (5) of the basic semidefinite problem (1) captures the nonlinearity of the problem in the eigenvalue function. The Spectral Bundle method `SB` from [25] is tailored to solve problems of the following type. For given $C, A_1, \ldots, A_m \in \mathcal{S}_n$ and $b \in \mathbb{R}^m$, define

$$f(y) = b^T y + \lambda_{\max}(C - \sum_i y_i A_i)$$

and consider

$$\min_{y \in \mathbb{R}^m} f(y).$$

In [25] it is pointed out that this problem is equivalent to the semidefinite program

$$\min\{b^T y : A^T y - C \succeq 0\}$$

in case that the identity matrix I is contained in the linear hull, spanned by the matrices A_i. It is clear that our problem (2) falls into this class of problems.

The spectral bundle method exploits the following easy facts. If $M \in \mathcal{S}^n$ then

$$\lambda_{\max}(M) = \max_{p^T p = 1} p^T M p = \max_{W \in \mathrm{conv}\{pp^T : p^T p = 1\}} \langle M, W \rangle = \max_{W \succeq 0, \mathrm{tr}(W) = 1} \langle M, W \rangle.$$

The bundle method generates a sequence of iterates that approaches the minimizer of $f(y)$. Suppose the current iterate is $\hat{y}$. Let $\hat{p}$ be a unit eigenvector to $\lambda_{\max}(C - A^T(\hat{y}))$. The main idea of the spectral bundle method consists in replacing the set $\{W : \mathrm{tr}(W) = 1, W \succeq 0\}$ by a suitable subset

$$S_P = \{W = PVP^T : \mathrm{tr}(V) = 1, V \succeq 0\},$$

defined through the 'bundle' P, which is an $n \times k$ matrix satisfying $P^T P = I_k$. The number k of columns in P can be selected by the user, and is typically much smaller than n. It is clear that

$$\lambda_{\max}(M) = \max_{W \in S_I} \langle M, W \rangle \geq \max_{W \in S_P} \langle M, W \rangle.$$

We define

$$\hat{f}(y) = b^T y + \max_{W \in S_P} \langle C - A^T(y), W \rangle \leq f(y).$$

The idea now is to select the matrix P in such a way that $\hat{f}(y)$ approximates $f(y)$ reasonably well around $\hat{y}$. We therefore assume that $\hat{p}$ is among the columns of P, ensuring that $f(\hat{y}) = \hat{f}(\hat{y})$. To make sure that the next iterate does not take us too far from the current iterate, the spectral bundle method minimizes $\hat{f}(y)$ incremented with a regularization term

$$\min_y \hat{f}(y) + \frac{1}{2t}\|y - \hat{y}\|^2.$$

The regularization parameter $t > 0$ is a user defined constant. Upon inserting the definition of $\hat{f}(y)$ we get a Min-Max problem

$$\min_y \max_{W \in S_P} b^T y + \langle C - A^T(y), W \rangle + \frac{1}{2t}\|y - \hat{y}\|^2.$$

In [25] it is shown that this can be solved by exchanging Min and Max, leading to the update

$$y = \hat{y} - t(b - A(PVP^T)),$$

with (yet) unknown V. To get V, one back-substitutes y to get the quadratic semidefinite program

$$\max_{W \in S_P} b^T \hat{y} + \langle C - A^T(\hat{y}), W \rangle - \frac{t}{2}\|b - A(W)\|^2,$$

where $W = PVP^T$. This quadratic semidefinite problem (with just one scalar equation $\mathrm{tr}(V) = 1$) can be solved by standard interior-point methods to yield V. This determines the new trial point

$$y_{\text{new}} = \hat{y} - t(b - A(PVP^T)).$$

The function f is now evaluated at the new point, the bundle P is updated (see [25] for details) and a new iteration is started. The main computational steps in one iteration are the solution of the quadratic semidefinite program to get V and the evaluation of the function at the new trial point, which amounts to an eigenvalue computation. A table with some representative results of this method applied to the dual problem (2) is reported in Section 6.

3.3 Non-Linear Programming Methods

In this section we review the Non-Linear Programming (NLP) based approaches for solving problem (1). Most of these algorithms exploit the special structure of the constraints of the problem in order to define non-linear programming reformulations.

We refer to a minimization problem, so that we define $Q = -C$ and we consider

$$\min\{\langle Q, X \rangle : \mathrm{diag}(X) = e,\ X \succeq 0\}. \tag{10}$$

This SDP relaxation of problem (3) was first derived by Goemans-Williamson in [15], by replacing each variable x_i of problem (3) with a vector $v_i \in \mathbb{R}^n$ (or $\mathbb{R}^r$ with $r \leq n$) obtaining problem (6). Since one can show the existence of a low rank optimal solution of (10) and any given matrix $X \succeq 0$ with rank r can be written as $X = VV^T$ for some $n \times r$ matrix V, the positive semidefiniteness

constraint can be eliminated, and problem (10) reduces to the so-called Low Rank SDP formulation (LRSDP)

$$\min_{V \in \mathbb{R}^{n \times r}} \{\langle Q, VV^T \rangle : \langle E_{ii}, VV^T \rangle = 1, i = 1, \ldots, n\}, \tag{11}$$

which turns out to be a non-convex non-linear problem (E_{ii} denotes the $n \times n$ matrix with the i-th diagonal component equal to one and all the other components equal to 0).

A global minimizer of problem (11) is a solution of problem (10) provided that r is not less than the rank $r_{\min}$ of a minimum rank optimal solution of problem (10). Although the value of $r_{\min}$ is not known, an upper bound can be easily computed by exploiting the result proved in [4, 21, 35], that gives

$$r_{\min} \leq \widehat{r} = \frac{\sqrt{8n+1}-1}{2}. \tag{12}$$

In principle, the value $\widehat{r}$ guarantees the correspondence between solutions of (10) and global solutions of (11).

However, even assuming that a suitable value of r is known, in order to solve problem (10), a global minimizer of the non-convex problem (11) is needed. In general this is a hard task, tied to the definition of suitable global optimality conditions. Thanks to strong duality and exploiting the connection between (1) and its dual (2), in [18, 19, 29] a global optimality condition has been proved. Indeed, let $r \geq r_{min}$. A feasible matrix $V^* \in \mathbb{R}^{n \times r}$ is a global minimizer of the LRSDP problem (11) if and only if

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

where $\lambda^* \in \mathbb{R}^n$ is uniquely defined component-wise as

$$\lambda_i^* = \lambda_i(V^*) = -\langle E_{ii} Q, V^* {V^*}^T \rangle \qquad i = 1, \ldots, n. \tag{13}$$

The LRSDP reformulation combined with the above optimality condition leads to the Incremental Rank Algorithm (IRA) for the solution of the SDP problem (10) reported below, that encompasses most of the algorithms proposed in the literature.

Incremental Rank Algorithm (IRA)

Data: Q and integer $p \geq 1$.
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 (12). Choose $\varepsilon > 0$.
For $j = 1, \ldots, p$
S.0 Set $r = r^j$ in problem (11), choose $V^0 = V^{0j} \in \mathbb{R}^{n \times r^j}$.
S.1 Starting from V^0, find a stationary point $\widehat{V} \in \mathbb{R}^{n \times r^j}$ of problem (11).
S.2 Compute the minimum eigenvalue $\lambda_{\min}(Q + \mathrm{Diag}(\lambda(\widehat{V})))$ of $Q + \mathrm{Diag}(\lambda(\widehat{V}))$.
S.3 If $\lambda_{\min}(Q + \mathrm{Diag}(\lambda(\widehat{V}))) \geq -\varepsilon$, then **exit**.
End
Return $\widehat{V} \in \mathbb{R}^{n \times r^j}$, $\lambda_{\min}(Q + \mathrm{Diag}(\lambda(\widehat{V})))$ and $z_{LB} = \langle Q, \widehat{V}\widehat{V}^T \rangle + n \min\{0, \lambda_{\min}(Q + \Lambda(\widehat{V}))\}$

The IRA scheme returns $\widehat{V}$, $\lambda_{\min}(Q + \mathrm{Diag}(\lambda(\widehat{V})))$ and z_{LB}. If $\lambda_{\min}(Q + \mathrm{Diag}(\lambda(\widehat{V}))) \geq -\varepsilon$, then the matrix $Q + \mathrm{Diag}(\lambda(\widehat{V}))$ is positive semidefinite within a tolerance ε so that a solution of (10) is obtained as $X^* = \widehat{V}\widehat{V}^T$. It is worth noting that, even if the optimality condition is not met, since $\left(\lambda(\widehat{V}) + \lambda_{\min}(Q + \mathrm{Diag}(\lambda(\widehat{V}))e\right)$ is feasible for the dual of problem (10), the value $z_{LB} = \langle Q, \widehat{V}\widehat{V}^T \rangle + n\lambda_{\min}\left(Q + \mathrm{Diag}(\lambda(\hat{V}))\right)$ provides a lower bound on the solution of problem (10) (see e.g, [19, 36] and the introduction of this chapter).

In principle, the IRA scheme may fail to produce a solution of problem (10) because the algorithm at Step S.1 may converge to stationary points which are not global minimizers of problem (11) even when $r = \widehat{r}$. Although this problem does not arise in practice, the drawback can be theoretically overcome by using an optimality result from [29]. Indeed, in that paper it has been proved that a rank deficient $\widehat{V}$ satisfying the second order necessary condition for problem (11) is necessarily a global minimizer. Hence, allowing an increase of r up to n and using an algorithm converging to second order points at Step S.1, the IRA scheme is guaranteed to converge to a global minimizer of problem (11).

What differentiates the various algorithms proposed in the literature is how to compute a stationary point of the LRSDP problem (11) at Step S.1, and whether the optimality check at Step S.2 is explicitly performed or not. In most papers, the computation of a stationary point is achieved by means of an unconstrained reformulation of problem (11).

In particular, in [8] the classical Augmented Lagrangian method [26, 37] is used for solving the LRSDP formulation of a general SDP problem with linear constraints. In particular, a sequence of minimizations of the Augmented Lagrangian function (specialized to problem (11))

$$\mathcal{L}(V, \lambda^k; \sigma^k) = \langle Q, VV^T \rangle + \sum_{i=1}^{n} \lambda_i^k \left(\langle E_{ii}, VV^T \rangle - 1 \right) + \frac{\sigma^k}{2} \sum_{i=1}^{n} \left(\langle E_{ii}, VV^T \rangle - 1 \right)^2 \tag{14}$$

is performed for suitable values of (λ^k, σ^k), where σ^k is increasing and λ^k is obtained with some updating rule.

In [19], Grippo et al. exploit the structure of the constraints in (11) to define an exact penalty function $P(V)$. The penalty function is obtained by replacing the multipliers λ in the augmented Lagrangian function $\mathcal{L}(V, \lambda; \sigma)$ with the closed expression (13), and by fixing the value of the penalty parameter to a computable value $\bar{\sigma} > 0$, so that $P(V) = \mathcal{L}(V, \lambda(V); \bar{\sigma})$. Thus, a single unconstrained minimization of the twice continuously differentiable function $P(V)$ provides a stationary point of problem (11).

More recently, Journée et al. [29] use a trust region method for the optimization over a manifold derived from [1] which relies on a particular quotient manifold. Their algorithm is defined for a slightly more general SDP problem than (10) but relies on exploiting the special structure of the constraints to find a closed expression of the multipliers which in the special case of problem (11) returns (13).

For the particular class of SDP problem (10) arising as a Max-Cut relaxation, Burer and Monteiro in the computational section of [8] use a different unconstrained formulation, based on a sort of Rayleigh quotient. The original idea goes back to Homer and Peinado [27], where the change of variables $x_{ij} = \langle v_i, v_j \rangle / \|v_i\| \|v_j\|$ 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 (11). Their approach leads to a problem of dimension n^2 which is solved by a parallel gradient method but turns out to be impractical for large values of n. Burer and Monteiro combine this approach with the low rank idea by replacing vectors $v_i \in \mathbb{R}^n$ with vectors $v_i \in \mathbb{R}^r$, for increasing values of r up to $\widehat{r}$, getting the unconstrained problem

$$\min \; f_r(V) = \sum_{i=1}^{n} \sum_{j=1}^{n} q_{ij} \frac{\langle v_i, v_j \rangle}{\|v_i\| \|v_j\|}, \qquad v_i \in \mathbb{R}^r. \tag{15}$$

The practical performance of this approach is pretty good, but the underlying convergence theory is not deeply investigated in [8]. It is easy to show the equivalence between the unconstrained problem (15) and problem (11) (see [16]). However, standard convergence results are not immediately applicable, since continuous differentiability of the objective function over the whole space $\mathbb{R}^{nr}$ is required, while the objective function of (15) is not even defined at points where $\|v_i\| = 0$ for at least one index i, and compactness of the level sets is missing as well. To overcome these drawbacks, Grippo et al. propose in [16, 17] a modification of the objective function of problem (15) which uses

an additive shifted barrier penalty term. The resulting unconstrained problem is

$$\min_{V \in S_\delta} f_r(V) + \tau \sum_{i=1}^{n} \frac{(\|v_i\|^2 - 1)^2}{\delta^2 - \max\{1 - \|v_i\|^2, 0\}^2} \tag{16}$$

where $S_\delta \equiv \{V \in \mathbb{R}^{n \times r} : \|v_i\|^2 > 1 - \delta, \quad i = 1, \ldots, n\}$, and $0 < \delta < 1$ and $\tau > 0$ are fixed scalars. Problem (16) has a continuously differentiable objective function on a suitable open set and compact level sets; moreover, equivalence to problem (11) still holds.

A performance comparison among all the approaches described here is presented in Section 6.

4 How to Find Good Max-Cut Solutions

As it was mentioned in the Section 1, an instance of Max-Cut defined by the weighted adjacency matrix $A \in \mathbb{R}^{n \times n}$ of a graph $G = (V, E)$ with $n = |V|$ nodes is equivalent to an instance of an *unconstrained binary quadratic program*

$$\max\{x^T Q x : x \in \{0, 1\}^{n-1}\}, \tag{17}$$

where Q is a symmetric matrix whose entries are determined from the matrix A, once a special node $s \in V$ has been selected and a correspondence between the nodes in $V \setminus \{s\}$ and the entries of x has been established (see [23]). A feasible solution to (17), i.e., an arbitrary binary $n-1$ vector, is readily mapped to a cut $\delta(S)$ of G, by defining S to be the set containing s and the nodes whose associated x entry is zero.

Due to the very simple structure of (17), it is relatively easy to design a heuristic algorithm for this problem and, as a matter of fact, many such algorithms, based on a variety of methods (evolutionary algorithms, simulated annealing, tabu-search, local search, etc.) have been proposed in the literature. One of the most recent references is, e.g., [6] which also contains pointers to many other papers.

In this section we only consider some heuristic algorithms that strongly rely on the solution of the SDP relaxation of Max-Cut.

One of the papers that have mainly contributed to making SDP techniques popular, in particular in the field of the approximation of hard combinatorial problems, is the one by Goemans and Williamson [15]. The paper describes and analyzes a randomized algorithm for Max-Cut that can be briefly outlined as follows.

Goemans-Williamson Algorithm

Data: C

S.1 Solve (1) and obtain an optimal solution X and the optimal value z_P.

S.2 Apply the Cholesky decomposition to the matrix X and obtain a matrix $V \in \mathbb{R}^{n\times r}$ such that $X = VV^T$. Let v_1, v_2, …, v_n be the row vectors of V.

S.3 From a uniform distribution, draw the components of a vector $h \in \mathbb{R}^r$.

Return z_P and the cut $\delta(S)$ where $S = \{i : \langle h, v_i\rangle \geq 0\}$.

We assume here that all the components of the weighted adjacency matrix of G are non-negative. The expected weight $\overline{W}$ of the cut is given by $\overline{W} = \sum_{ij\in\delta(S)} A_{ij}p_{ij}$, where p_{ij} is the probability that 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 $\langle h, x\rangle = 0$. 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

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

In conclusion, the expected 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 $O(n^3)$ operations (so that computational time is proportional to n^3) and needs memory space proportional to n^2. Therefore, the algorithm cannot be applied to very large instances with size, say, on the order of one hundred thousand nodes.

By exploiting a particular implementation of the IRA scheme described in Section 3.3, in [17] an efficient heuristic for finding good solutions for very large graphs has been proposed. Indeed, an efficient implementation of the IRA scheme makes it possible to apply the Goemans and Williamson approximation algorithm to very large graphs, since on the one hand it is able to solve problem (10) in reasonable time also for very large graphs, and on the other hand, it outputs (for free) the vectors v_i, avoiding the need of a Cholesky factorization. The cut provided by the Goemans and 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. In [12], a particularly successful step is to re-apply the heuristic starting from a convex combination of the solution X of the relaxation and of the rank one matrix $\hat{x}\hat{x}^T$, where $\hat{x}$ is the current best cut. This procedure aims at moving

towards a 'good vertex'. If the graph is too large, however, this is not practical since it requires the Cholesky factorization of the matrix $\alpha X + (1-\alpha)\hat{x}\hat{x}^T$. In order to move towards a 'good vertex' without using any Cholesky factorization, in [17] a perturbation C' of the original objective function matrix C in (1) is applied, that is given by $C' = C - \beta\hat{x}\hat{x}^T$ with $\beta > 0$. Such a perturbation has the same effect of moving toward a 'good vertex' as by increasing β, the weights of the edges in the cut defined by $\hat{x}$ get increased, while the weights of those outside of this cut get decreased. The SDP problem (1) with $C = C'$ is then solved by again using the IRA scheme and the whole heuristic is repeated a given number of times. Summarizing, the scheme of the heuristic algorithm, proposed in [17], is the following:

IRA-based Heuristic

Data: $C,\ \gamma > 0,\ k_{\max}$.
Initialization: Set $\overline{Q} = \sum_{i,j} |C_{ij}|/|E|,\ \hat{x} = e$.
For $k = 0, \ldots, k_{\max}$:
- **S.0** Set $\beta^k = (k_{\max} - k)\gamma\overline{Q}$
- **S.1** Apply **IRA** to problem (10) with $Q = -C + \beta^k\hat{x}\hat{x}^T$, and let v_i, $i = 1, \ldots, n$, be the returned solution and $-z_{LB}$ a valid bound for the corresponding Max-Cut problem.
- **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** Locally improve the cut $\bar{x}$ by checking all possible moves of a single vertex to the opposite set of the bipartition. This gives a new bipartition representative vector $\widetilde{x}$.
  If $\langle C, \widetilde{x}\widetilde{x}^T\rangle > \langle C, \hat{x}\hat{x}^T\rangle$, set $\hat{x} = \widetilde{x}$.

End
Return Best cut $\hat{x}$, lower bound $\langle C, \hat{x}\hat{x}^T\rangle$, upper bound $-z_{LB}$.

Note that the perturbation decreases when the iteration counter increases, getting to zero in the last iteration. On the other hand, in the first iteration the whole matrix is perturbed. We stress that, after the first iteration, Step S.1 is not expensive since a warm start technique can be used: at each iteration IRA is started by the solution found at the previous step, so that the new minimization is computationally cheap. Numerical results for this heuristic are reported in Section 6. Another advantage of this heuristic is that it also provides a performance measure, since it computes both an upper and lower bound on the value of the optimal cut.

5 How to Find Optimal Max-Cut Solutions

As mentioned in Section 1, Max-Cut is an NP-hard problem and requires using some pseudo-enumeration techniques in order to be solved to optimality. In a

typical solution algorithm, an instance of the problem is recursively replaced by two other instances obtained by imposing that two nodes of the graph belong to the same set or to two distinct sets of the bipartition, respectively.

Given a graph $G = (V, E)$ with weighted adjacency matrix A and a selected node s, by $\langle A, U, W\rangle$ we denote the Max-Cut instance defined on G but with the additional constraints that the nodes of the set $U \subseteq V$ have to be in the same set of the bipartition as s and those of a set $W \subseteq V$ have to be in the other set.

It is interesting to note that $\langle A, U, W\rangle$ is the instance of a Max-Cut problem in a smaller graph with no additional constraints. This is easily seen from the formulation (17) of the problem. Consider the two instances that are obtained from a given one by imposing that a variable x_i has to be equal to 0 or to 1, respectively. We obtain two problems again of the form (17) with one less variables and with matrices Q_0 and Q_1, respectively. Matrix Q_0 is obtained from Q by deleting the i-th row and column, while Q_1 is obtained from Q_0 by suitably modifying its diagonal. In the latter case a constant has also to be added to the objective function as well.

We call $M(\langle A, U, W\rangle)$ the weighted adjacency matrix of the graph that corresponds to the instance $\langle A, U, W\rangle$ and $k(\langle A, U, W\rangle)$ the constant that has to be added to the weight of a cut in this graph to obtain the weight of the corresponding cut in G. The size of $M(\langle A, U, W\rangle)$ is $|V| - |U| - |W| + 1$, thus the larger the sets U and W, the smaller (and simpler) is the corresponding Max-Cut instance.

The process of replacing a problem by two simpler subproblems can be represented by a binary tree. The growing of the tree from a node is interrupted as soon as the *upper bound*, i.e., the value of a bound on the Max-Cut optimal value for the instances corresponding to that node (found, e.g., by solving (1)) falls below the *lower bound*, i.e., the weight of the best available cut (found, e.g., with the heuristic techniques of Section 4). If this never happens, the procedure produces a tree of 2^{n-1} nodes which amounts to solving the problem by complete enumeration.

The above scheme is computationally acceptable only if the number of nodes is kept relatively small, which is obtained by providing good heuristics and good relaxations.

5.1 LP Relaxations

A popular and successful relaxation for Max-Cut is based on Linear Programming techniques. An alternative way to encode a cut is by a $0-1$ vector in $\mathbb{R}^E$ with component, corresponding to the edge connecting the nodes i and j, equal to 1 if the edge ij belongs to the cut and 0 otherwise. The convex hull of the encoding vectors of all cuts of G is the Cut Polytope $\mathcal{C}(G)$. Then solving Max-Cut is equivalent to optimizing a linear function over $\mathcal{C}(G)$. Due to the enormous number of inequalities that are needed to define this polytope (most of which are unknown), one has to use simpler and larger polytopes.

A polytope containing $\mathcal{C}(G)$, that can thus be used to obtain a relaxation of Max-Cut, is the Semimetric Polytope $\mathcal{M}(G)$ which is defined by the following inequalities

$$\begin{aligned} &\sum_{ij\in F} z_{ij} - \sum_{ij\in C\setminus F} z_{ij} \leq |F| - 1 && \text{for all cycles } C \text{ of } G \text{ and} \\ &&& \text{for all } F \subseteq C \text{ with } |F| \text{ odd} \\ &0 \leq z_{ij} \leq 1 && \text{for all } ij \in E \end{aligned} \tag{18}$$

that encode the property that the intersection of a cut with a cycle of G always has an even cardinality. For this reason the inequalities in the first set of (18) are called the *cycle inequalities*.

In general the number of cycle inequalities is exponential in n. However, optimizing a linear function over $\mathcal{M}(G)$ is still efficiently doable with a *cutting plane algorithm*. One starts by optimizing a linear program defined by a very small subset of (18) and then checks if the optimal solution satisfies the whole set. In the affirmative case the algorithm stops, otherwise it adds some violated inequalities to the formulations and iterates the process. Despite the exponential size of the set of cycle inequalities, solving the *separation problem* for the cycle inequalities, i.e., finding one which is violated by a given point, can be done in polynomial time [3]. This fact, combined with the efficiency of today's linear program solvers, makes one expect a fast computation of the semimetric bound. Unfortunately, this is true only for very sparse graphs (like grid graphs where the number of edges is only twice the number of nodes). For these graphs using the semimetric bound makes it possible to optimize instances of several thousand nodes in a reasonable amount of time (see, e.g., [33]).

But what about denser instances? In this case a stronger relaxation would be necessary which could be obtained by enriching the set of the valid inequalities that are used in the above cutting plane algorithm. Several families of valid or even facet defining inequalities for $\mathcal{C}(G)$ have been characterized and thus provide a source for possible stronger relaxations.

Observe also that a cycle inequality is not facet defining, and thus is not essential in the formulation of a relaxation, if the cycle contains a chord. In addition the trivial inequalities $0 \leq z_{ij} \leq 1$ are not facet defining if the edge ij is contained in a triangle. Therefore, if G is a complete graph with n nodes, denoted by K_n, the corresponding Semimetric Polytope $\mathcal{M}(K_n)$ is completely described by cycle inequalities where each cycle has length 3. Such inequalities are also called the *triangle inequalities*.

A detailed treatment on most of the families of facet defining inequalities for $\mathcal{C}(G)$ known to date is contained in the book [10]. Unfortunately, the separation problem is difficult for most of them. In addition, many such inequalities are defined only for the cut polytope on a complete graph and it is not easy to exploit them when the graph is not complete. Because of these difficulties, no computational studies are available today, where the use of an enriched set of inequalities is the key for solving difficult dense instances.

5.2 SDP Relaxations

A breakthrough in the exact solution of dense instances was made when the SDP relaxation was introduced: instances on complete graphs of 50 to 100 nodes, very difficult for Linear Programming techniques, could be solved in very reasonable amount of time by exploiting the SDP bound in the above mentioned pseudo-enumeration scheme.

Unfortunately, the bound provided by (1) is not particularly strong. For example, using that bound, an instance of 100 nodes can be solved in about 20 minutes on a laptop after the generation of 30 000 nodes. These numbers tend to increase very rapidly as n increases. Therefore, instances of slightly larger size remain practically unsolvable if one only relies on the bound given by (1).

To generate stronger bounds several formulations have been proposed, building on (1) by enlarging (even substantially) the dimension of the solution space (see e.g. [2, 31]). However, these formulations do not appear, at the moment, to be of practical use for instances of 100 or more nodes.

We said that for denser graphs the SDP bound outperforms the semimetric bound in terms of computation time. But what about the qualities of the two bounds? Is there one of the two that is always preferable to the other? The answer is unfortunately no: either one can be the winner, depending on the weighted adjacency matrix of the graph.

The projection of the feasible set of (1) onto the space of the off-diagonal and upper-triangular components of X is a convex body, called *elliptope* or Max-Cut *spectrahedron* (see e.g. [32]), that contains the cut polytope $\mathcal{C}(K_n)$. Therefore, a very natural way to improve the SDP bound would be to optimize over the intersection of the elliptope with the semimetric polytope or with any stronger relaxation obtained by adding inequalities, valid for $\mathcal{C}(K_n)$, to the triangle inequalities.

5.3 Combining LP and SDP relaxations

Two ways of optimizing over this intersection have been experimented. The first is described in [24], where, at every iteration of the bound computation, the triangle inequalities are added to the formulation (1), like it is done in the linear programming based cutting plane algorithms. The resulting SDP problem is solved with an interior-point algorithm. The bound computation is then embedded into a pseudo-enumeration scheme. The resulting algorithm is able to solve instances up to 100 nodes. However, the SDP computation becomes slower and slower as long as new inequalities are added. Therefore, the approach becomes computationally impractical for graphs of more than 100 nodes.

A different approach is adopted in [40] where a branch and bound algorithm, named BiqMac, is described. The bound is computed iteratively by adding triangle inequalities to the formulation as it is done in cutting plane algorithms. However, differently from the previous method, the feasible set

of (1) is left unchanged, while the objective function is modified at every iteration by dualizing all the identified inequalities in a Lagrangian fashion. The dual functional is then optimized using the bundle method.

The algorithm can be outlined as in Figure 1, where C denotes the weighted adjacency matrix of a graph with node set V, edge set E and weights given by a triangle inequality, $r(C)$ denotes its right hand side and $\gamma(C)$ the dual variable associated with the inequality.

Data: A complete graph $G = (V, E)$, A weighted adjacency matrix, $s \in V$.
Initialization: Set the problem list $\mathcal{P} = \{\langle A, \{s\}, \emptyset\rangle\}$, $lb = 0$, and $S = \emptyset$.
While $\mathcal{P} \neq \emptyset$ **do**
S.0 Extract $\langle A, U, W\rangle$ from $\mathcal{P}$.
S.1 Compute the Laplacian matrix L associated with the matrix $M(\langle A, U, W\rangle)$ and set $\mathcal{C} = \emptyset$.
S.2 **While** Condition (A) is satisfied **do**
S.2.0 Define the Lagrangian

$$\mathcal{L}(X, \gamma) = \langle L, X\rangle + \sum_{C\in\mathcal{C}} \gamma(C) r(B) - \sum_{C\in\mathcal{C}} \gamma(C)\langle C, X\rangle.$$

S.2.1 Define the dual functional

$$f(\gamma) = \max\{\mathcal{L}(X, \gamma) : \mathrm{diag}(X) = e, X \succeq 0\}.$$

S.2.2 Solve

$$z_{\mathrm{mc-met}} = \min\{f(\gamma) : \gamma \geq 0\}$$

by the bundle method, using SDP optimization to evaluate $f(\gamma)$. Let $\tilde{X}$ be the optimal solution.
S.2.3 Solve the separation problem for the triangle inequalities relative to the point $\tilde{X}$. If any inequalities are found, for each of them add the corresponding matrix C to the set $\mathcal{C}$.
end
S.3 Using the last solution $\tilde{X}$ computed, find a 'good' cut using the Goemans-Williamson algorithm, possibly followed by some local improvement procedure. Let this cut be defined by the set $\overline{S}$, with $s \in \overline{S}$ and let $\overline{z}$ be its value.
If $\overline{z} + k(\langle A, U, W\rangle) > lb$, then set $lb = \overline{z} + k(\langle A, U, W\rangle)$ and $S = \overline{S} \cup U$.
S.4 If $z_{\mathrm{mc-met}} + k(\langle A, U, W\rangle) > lb$, then select a node $t \in V \setminus (U \cup W \cup \{s\})$ and add the two instances $\langle A, U \cup \{t\}, W\rangle$ and $\langle A, U, W \cup \{t\}\rangle$ to $\mathcal{P}$.
end
Return S and lb.

Fig. 1. The BiqMac Algorithm

Two technical details in the algorithm described in Figure 1 need to be clarified. The first concerns Condition (A) in Step S.2. This condition is satisfied as long as the separation problem finds some triangle inequalities violated and as long the process of adding more inequalities to the set $\mathcal{C}$ produces a satisfactory decrease in the value $z_{\mathrm{mc-met}}$. Observe that, as the triangle inequalities are $4\binom{n}{3}$, the separation procedure trivially amounts to checking each of them for violation. The other detail concerns the way node t is chosen in Step S.4. There are several rules, the easiest being the one where the chosen node corresponds to the entry of $\tilde{X}$ with smallest absolute value in the row of node s.

6 Computational Status

In this section, we explore the actual computational status for solving the SDP problem (1), and for finding good cuts for the Max-Cut problem (3). Furthermore, we report a summary of the computational behavior of the BiqMac algorithm.

As an interior-point method we select the dual-scaling algorithm defined in [5] and implemented in the software `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. Furthermore, `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` of [25] described in Section 3.2.

Among the non-linear programming based methods, we consider the different approaches described in Section 3.3 for solving problem (11) as required at Step S.1 of the IRA scheme. As mentioned in Section 3.3, in [8] two different unconstrained formulations have been proposed for the solution of the non-linear problem (11) and have been implemented in two C codes. The first one is a general purpose code for solving linear SDPs based on the sequential Augmented Lagrangian methods with the minimization performed by means of a limited memory L-BFGS method. The corresponding IRA algorithm is implemented in the code `SDPLR` which does not include explicitly Step S.2. The IRA scheme for the special structured problem (10) based on the second unconstrained formulation is implemented in the code `SDPLR-MC`. This code is based on the unconstrained formulation (15) solved by means of an L-BFGS algorithm, with the updating rule for the values of the rank $r^j = \lceil \frac{j}{p}\widehat{r} \rceil$ ($\widehat{r}$ given by (12) and $p = 5$). The computational results in [8] show that the method `SDPLR-MC` outperforms `SDPLR`, thus we include in our comparison only `SDPLR-MC`. We remark again that both `SDPLR` and `SDPLR-MC` do not certify global optimality of the produced solution for problem (1) in the sense that they do not check the global optimality condition $Q + \mathrm{Diag}(\lambda(\hat{V})) \succeq 0$ at Step S.2 of the IRA scheme.

The manifold optimization method GenRTR defined in [29] is implemented in Matlab, so that direct comparison in terms of computational time with other C, C++ or Fortran codes is not really fair. However, in [29] the authors report a comparison on some Max-Cut instances of GenRTR and SDPLR. The reported results show that the two methods may be considered to have comparable performances. This implies that on the special structured problem (1) that comes from Max-Cut, GenRTR should have worse performances than SDPLR-MC, so that we do not include GenRTR in our comparison.

The IRA algorithm based on the exact penalty approach proposed in [19] is implemented in the code EXPA where the minimization of the exact penalty $P(V)$ is performed by a non-monotone Barzilai-Borwein gradient method [20]. In [16] this same method is used for solving the unconstrained problem (16). The corresponding IRA algorithm is implemented in the code SpeeDP with the updating rule of the rank $r^{j+1} = \min\{1.5 \times r^j, \widehat{r}\}$ $j = 1, 2 \ldots$, with a dimension based heuristic for the choice of the value r^1. We remark that both EXPA and SpeeDP include the evaluation of the minimum eigenvalue at Step S.2 of the IRA scheme using the ARPACK subroutines *dsaupd* and *dseupd*.

Thus, to summarize, we report comparison among DSDP, SB, SDPLR-MC, EXPA and SpeeDP as done in [17] where an extensive numerical comparison of the performance of the different codes on a test bed consisting of 38 widely used instances of Max-Cut is presented. To give a flavor of the behavior of the methods, we report here only the results on six selected problems with an increasing dimension ranging from $|V| = 250$ to $|V| = 10,000$ and from $|E| \cong 1300$ to $|E| \cong 30,000$ (mcp250-3, mcp500-3, G51 from the SDPLIB collection http://euler.nmt.edu/∼brian/sdplib/sdplib.html and G35, G58, G72 from the Gset collection http://dollar.biz.uiowa.edu/∼burer/software/SDPLR).

We compare the performance of the different codes in term of computational time and accuracy. The CPU time, in seconds, on each instance are reported in Table 1 together with the dimension $|V| = n$ and $|E|$.

graph	$\|V\|$	$\|E\|$	SpeeDP	EXPA	SDPLR-MC	SB	DSDP
1	250	1283	0.04	0.15	0.14	0.45	0.52
2	500	2355	0.11	0.52	0.43	0.98	2.40
3	1000	5909	0.24	3.07	3.10	6.74	15.70
4	2000	11778	1.45	5.03	10.22	31.13	117.00
5	5000	29570	15.74	77.94	45.68	368.23	1969.00
6	10000	20000	8.21	35.77	32.92	1471.95	6017.00

Table 1. CPU time of the three NLP based methods, SB and DSDP

It emerges that the time increases with the dimension, as expected. The non-linear programming based methods are faster than both the interior-point and spectral bundle methods. Among the non-linear programming based

methods, SpeeDP shows the most competitive performance. This is true not only on these six problems but on the overall test bed of 38 problems. Complete tables related to these numerical tests can be found in [17]. To give a

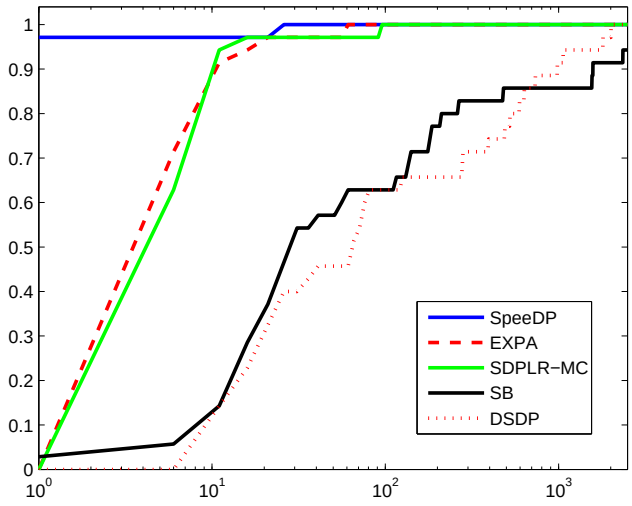

Fig. 2. Comparison between NLP based methods, SB and DSDP (the x-axis is the CPU time in seconds reported in logarithmic scale)

view of the behavior with respect to the computational time on the whole test bed (except the two largest problems G77 and G81 of the Gset collection where DSDP runs out of memory), we draw the performance profile [11] in Figure 2. We recall that the higher a method's profiles appears, the better is the method's performance.

As for the accuracy, a comparison among primal and/or dual objective function values obtained by the five methods is required. We first observe that in the non-linear programming methods falling within the IRA scheme (SDPLR-MC, EXPA, SpeeDP), the primal objective value $\langle Q, \widehat{V}\widehat{V}^T\rangle$ is always reported on exit together with a measure of dual infeasibility $\lambda_{\min}(Q+\Lambda(\widehat{V}))$. Whenever $\lambda_{\min}(Q+\Lambda(\widehat{V})) \geq 0$ perfect dual feasibility is obtained, so that the value of the primal function gives a valid bound for the Max-Cut problem (3). Whenever this is not true a valid bound, namely the value of the dual function, is obtained as $\langle Q, \widehat{V}\widehat{V}^T\rangle + n \min\{0, \lambda_{\min}(Q+\Lambda(\widehat{V}))\}$. We note that the codes EXPA and SpeeDP give this value as an output, whereas SDPLR-MC only reports the primal objective value. Interior-point methods report both primal and dual values, while the spectral bundle method SB produces a value of the dual objective function that is a bound on the optimal value of problem (1).

By analyzing the results on the whole test bed (see tables and figures in [17]), it emerges that non-linear programming methods and SB methods have usually worse accuracy than interior-point methods. `SpeeDP` is an exception, since its accuracy can be considered comparable with interior-point methods. Indeed `SpeeDP` finds perfect dual feasibility on 20 problems, it is more accurate than `DSDP` on 8 problems (comparing the dual objective function) and, on the 10 problems where `SpeeDP` is less accurate, the difference is less than 10^{-4}. This comparable level of accuracy is obtained with a significant saving of time when the dimension of the problem increases.

Now, we consider the state-of-the-art method for solving the Max-Cut problem to optimality, the BiqMac algorithm described in Section 5.3. BiqMac solves any instance of size up to $n = 100$ routinely and larger instances of a special structure up to $n = 300$. An extensive computational test can be found in [40], where the solution of some still unsolved instances is reported. A server based on this algorithm, to which one can submit Max-Cut as well as unconstrained binary quadratic instances is accessible at the site [39] or via the NEOS Server for Optimization.

Finally, we recall some numerical results obtained by a heuristic that fits within the scheme described in Section 4. Indeed, in [17] the heuristic `SpeeDP-MC`, based on the use of the `SpeeDP` algorithm within the scheme of Section 4, has been proposed. In [17] `SpeeDP-MC` has been tested on large random graphs, with number of nodes between $n = 500$ and $n = 2500$ for edge densities varying between 10% and 100% and weights drawn from different distributions. It emerges from the tests that 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 (intended as the distance between the solution found and the bound produced) decreases when the density of the graph increases since the quality of the bound improves. We consider it worth mentioning here that in [17] `SpeeDP-MC` has been used to compute good feasible solutions of the Max-Cut problem on some huge 6-regular random graphs (3D toroidal grid graphs) with millions of nodes and edges and different weight ranges. These results, reported in Table 2, were surprisingly good.

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 2. 6-regular graphs with 1 030 301 nodes and 3 090 903 edges

Summarizing, we believe that the efficiency of non-linear programming based methods may be successfully used within heuristic and/or exact schemes for solving Max-Cut. Methods in other classes (such as interior-point or spectral bundle methods) turn out to be extremely accurate and more naturally extendable to classes of SDP problems more general than problem (1).

7 Conclusions

In this chapter, we surveyed some recent results on the Max-Cut problem, focusing mainly on the impact that the progress in SDP development has on the computation of good/optimal solutions of this problem. Since the Max-Cut problem is NP-hard, some pseudo-enumeration technique is needed to solve it exactly. Therefore the computation of good upper and lower bounds is fundamental for any solution algorithm for the Max-Cut problem, and the last years of research showed that the simple SDP problem (10) plays a fundamental role in all the most efficient algorithms. This SDP problem is also contained in the relaxations arising from some related problems: Max-k-Cut, Coloring and Ordering problems described in Section 2. In the chapter we surveyed the methods available for solving this SDP problem: interior-point methods, the spectral bundle method and a recent class of non-linear programming based methods, that have a huge computational impact on the ability of solving large sparse instances. A method belonging to this class has been exploited to define a heuristic algorithm which is able to find good solutions for huge instances of Max-Cut as described in Section 4. Furthermore, we described BiqMac, the best SDP based method available in the literature for solving the Max-Cut problem to optimality. We concluded the chapter by a computational section where all the methods described are compared and where a summary of the numerical results shows the big impact of the new non-linear programming based methods on the size of the instances of Max-Cut that can be tackled.

Acknowledgments

We thank the two anonymous referees for their careful reading of the chapter, and for their constructive remarks that greatly helped to improve its presentation.

References

1. P.-A. Absil, C.G. Baker, and K.A. Gallivan. Trust-region methods on Riemannian manifolds. *Journal Foundations of Computational Mathematics*, 7(3):303–330, 2007.
2. M.F. Anjos and H. Wolkowicz. Strengthened semidefinite relaxations via a second lifting for the Max-Cut problem. *Discrete Applied Mathematics*, 119(1-2):79–106, 2002.
3. F. Barahona and A.R. Mahjoub. On the cut polytope. *Mathematical Programming*, 36:157–173, 1986.
4. A. Barvinok. Problems of distance geometry and convex properties of quadratic maps. *Discrete Computational Geometry*, 13:189–202, 1995.
5. S.J. Benson, Y. Ye, and X. Zhang. Solving large-scale sparse semidefinite programs for combinatorial optimization. *SIAM Journal on Optimization*, 10(2):443–461, 2000.
6. E. Boros, P.L. Hammer, and G. Tavares. Local search heuristics for quadratic unconstrained binary optimization. *Journal of Heuristics*, 13:99–132, 2007.
7. C. Buchheim, A. Wiegele, and L. Zheng. Exact algorithms for the Quadratic Linear Ordering Problem. *INFORMS Journal on Computing*, 22(1):168–177, 2010.
8. S. Burer and R.D.C. Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical Programming*, 95:329–357, 2003.
9. E. de Klerk, D.V. Pasechnik, and J.P. Warners. On approximate graph colouring and Max-k-Cut algorithms based on the ϑ-function. *Journal of Combinatorial Optimization*, 8(3):267–294, 2004.
10. M. Deza and M. Laurent. *Geometry of Cuts and Metrics*, volume 15 of *Algorithms and Combinatorics*. Springer-Verlag, Berlin, 1997.
11. E.D. Dolan and J.J. Morè. Benchmarking optimization software with performance profile. *Mathematical Programming*, 91:201–213, 2002.
12. 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. *Mathematical Programming*, 105(2-3, Ser. B):451–469, 2006.
13. A. Frieze and M. Jerrum. Improved approximation algorithms for Max k-Cut and Max Bisection. *Algorithmica*, 18(1):67–81, 1997.
14. B. Ghaddar, M. Anjos, and F. Liers. A branch-and-cut algorithm based on semidefinite programming for the minimum k-partition problem. *Annals of Operations Research*, pages 1–20, 2008.
15. M.X. Goemans and D.P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the Association for Computing Machinery*, 42(6):1115–1145, 1995.
16. L. Grippo, L. Palagi, M. Piacentini, and V. Piccialli. 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.
17. L. Grippo, L. Palagi, M. Piacentini, V. Piccialli, and G. Rinaldi. SpeeDP: A new algorithm to compute the SDP relaxations of Max-Cut for very large graphs. Technical Report 13.10, DII-UTOVRM - Università di Roma Tor Vergata, 2010. available on Optimization Online.

18. 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(3):339–348, 2009.
19. L. Grippo, L. Palagi, and V. Piccialli. 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.
20. L. Grippo and M. Sciandrone. Nonmonotone globalization techniques for the Barzilai-Borwein gradient method. *Computational Optimization and Applications*, 23:143–169, 2002.
21. R. Grone, S. Pierce, and W. Watkins. Extremal Correlation Matrices. *Linear Algebra Application*, 134:63–70, 1990.
22. M. Grötschel, M. Jünger, and G. Reinelt. Facets of the linear ordering polytope. *Mathematical Programming*, 33:43–60, 1985.
23. P. Hammer. Some network flow problems solved with pseudo-boolean programming. *Operations Research*, 13:388–399, 1965.
24. C. Helmberg and F. Rendl. Solving quadratic $(0,1)$-problems by semidefinite programs and cutting planes. *Mathematical Programming*, 82(3):291–315, 1998.
25. C. Helmberg and F. Rendl. A spectral bundle method for semidefinite programming. *SIAM Journal on Optimization*, 10:673–696, 2000.
26. M. Hestenes. Multiplier and gradient methods. *Journal of Optimization Theory and Application*, 4:303–320, 1969.
27. S. Homer and M. Peinado. Design and performance of parallel and distributed approximation algorithm for the Maxcut. *Journal of Parallel and Distributed Computing*, 46:48–61, 1997.
28. P. Hungerländer and F. Rendl. Semidefinite relaxations of ordering problems. Technical report, Alpen-Adria Universität Klagenfurt, Austria, 2010.
29. M. Journée, F. Bach, P.A. Absil, and R. Sepulchre. Low-rank optimization for semidefinite convex problems. *SIAM Journal on Optimization*, 20(5):2327–2351, 2010.
30. D. Karger, R. Motwani, and M. Sudan. Approximate graph colouring by semidefinite programming. *Journal of the Association for Computing Machinery*, 45:246–265, 1998.
31. J.B. Lasserre. An explicit equivalent positive semidefinite program for nonlinear 0-1 programs. *SIAM Journal on Optimization*, 12(3):756–769, 2002.
32. M. Laurent and F. Rendl. Semidefinite programming and integer programming. In K. Aardal, G.L. Nemhauser, and R. Weismantel, editors, *Discrete Optimization*, volume 12 of *Handbook in OR & MS*, chapter 8. Elsevier B.V., 2005.
33. F. Liers, M. Jünger, G. Reinelt, and G. Rinaldi. Computing exact ground states of hard Ising spin glass problems by branch-and-cut. In A.K. Hartmann and H. Rieger, editors, *New Optimization Algorithms in Physics*, pages 47–69. Wiley-VCH Verlag, 2004.
34. L. Lovász. On the Shannon capacity of a graph. *IEEE Transactions on Information Theory*, IT–25:1–7, 1979.
35. G. Pataki. On the rank of extreme matrices in semidefinite programs and the multiplicity of optimal eigenvalues. *Mathematics of Operations Research*, 23:339–358, 1998.
36. S. Poljak, F. Rendl, and H. Wolkowicz. A recipe for semidefinite relaxation for 0-1 quadratic programming. *Journal of Global Optimization*, 7:51–73, 1995.
37. M.J.D. Powell. *Optimization*, chapter A method for nonlinear constraints in minimization problem, pages 283–298. Academic Press, New York, 1969.

38. G. Reinelt. *The Linear Ordering Problem: Algorithms and Applications*. Heldermann Verlag, 1985.
39. F. Rendl, G. Rinaldi, and A. Wiegele. Biq Mac Solver - BInary Quadratic and MAx-Cut Solver. http://biqmac.uni-klu.ac.at/.
40. F. Rendl, G. Rinaldi, and A. Wiegele. Solving Max-Cut to optimality by intersecting semidefinite and polyhedral relaxations. *Mathematical Programming*, 121:307–335, 2010.