Prof. Daniel Kressner

Supervisor: Petar Sirkovic

Eigenvalue Optimization for Solving the MAX-CUT Problem

Julien Hess

Lausanne, fall 2012
Abstract

The purpose of this semester project is to investigate the Spectral Bundle Method, which is a specialized subgradient method particularly suited for solving large scale semidefinite programs that can be cast as eigenvalue optimization problems of the form

\[
\min_{y \in \mathbb{R}^m} a \lambda_{\text{max}}(C - \sum_{i=1}^{m} A_i y_i) + b^T y,
\]

where \(C\) and \(A_i\) are given real symmetric matrices, \(b \in \mathbb{R}^m\) allows to specify a linear cost term, and \(a > 0\) is a constant multiplier for the maximum eigenvalue function \(\lambda_{\text{max}}(\cdot)\). In particular, a semidefinite relaxation of the well-known MAX-CUT problem belongs to this class of problems.

After a general description of the Spectral Bundle Method, a MATLAB implementation of the method designed for solving the eigenvalue relaxation of the MAX-CUT problem is presented. Finally, it is explained how to extract an optimal solution of the original MAX-CUT problem from the optimal solution of the eigenvalue relaxation.
## Contents

1 Introduction 4

2 Preliminaries 5
   2.1 Notation ........................................ 5
   2.2 Linear Algebra ................................. 6

3 The Max-Cut Problem 7
   3.1 Standard Formulation .......................... 7
   3.2 Algebraic Formulation .......................... 8

4 Semidefinite Programming 10
   4.1 The Standard Primal-Dual Pair of Semidefinite Programs ....... 10
   4.2 Duality Theory ................................ 11
   4.3 Semidefinite Relaxation of the Max-Cut Problem ............... 12
   4.4 Solving the Semidefinite Relaxation with SeDuMi .......... 13

5 Eigenvalue Optimization 14
   5.1 Standard Formulation .......................... 14
   5.2 Eigenvalue Relaxation of the Max-Cut Problem ............... 15
   5.3 Analysis of the Objective Function .................. 16
   5.4 The Spectral Bundle Method .................... 18
      5.4.1 Verbal Description ......................... 18
      5.4.2 Mathematical Description .................. 19
      5.4.3 The Algorithm ............................. 24
      5.4.4 Improvements .............................. 25

6 Resolution of the Max-Cut Problem 26
   6.1 Extraction of a Dual Optimal Solution ................ 26
   6.2 Extraction of a Primal Optimal Solution ............. 26
   6.3 Best Rank One Approximation ..................... 28
   6.4 Extraction of an Optimal Cut ..................... 28

7 Numerical Experiments 29

A Matlab codes 34
   A.1 The Spectral Bundle Method ..................... 34
      A.1.1 Lanczos with Reorthogonalization .......... 34
      A.1.2 The Oracle ............................... 35
      A.1.3 The Trial Point Finding ................... 35
      A.1.4 The Model Update .......................... 36
      A.1.5 The Algorithm ............................. 36
   A.2 Resolution of the Max-Cut Problem ............... 38
      A.2.1 Extraction of a Dual Optimal Solution .... 38
      A.2.2 Extraction of a Primal Optimal Solution ... 38
      A.2.3 Best Rank One Approximation .............. 39
      A.2.4 Extraction of an Optimal Cut ............. 39
1 Introduction

Although it is one of the simplest graph partitioning problems to formulate, the max-cut problem (MC) is one of the most difficult combinatorial optimization problems to solve. Indeed, it belongs to the class of NP-complete problems, meaning that it is not yet known whether it can be solved with an algorithm running in polynomial time for general graphs. For this reason, it is customary to solve a relaxation of the problem rather than MC itself, which provides an upperbound for the optimal solution.

The most common relaxation of the max-cut problem is expressed in the framework of semidefinite programming. From a theoretical point of view, interior point methods are ideal for solving semidefinite programs (see Helmberg [3], Chap. 4). However, for practical applications with many constraints, the price to pay in a single iteration is often too high. In this view, the Spectral Bundle Method - which is a specialized subgradient method - offers an interesting alternative for solving the class of large scale semidefinite programs that can be cast as eigenvalue optimization problems (cf. [3], Chap. 5). As the max-cut problem belongs to this class, the main purpose of this paper is to show how the Spectral Bundle Method can be used in order to solve it.

To begin with, we introduce the notation used in this paper and give some basic results of linear algebra in Section 2. In Section 3 we introduce the max-cut problem in more detail, whereas the standard formulation of the primal-dual pair of semidefinite programs and the main results of duality theory are briefly presented in Section 4. The reader familiar with the max-cut problem and with semidefinite programming might want to skip those sections and go directly to Section 5, where the main topic of this paper is discussed. A general description of the Spectral Bundle Method is given, as well as a Matlab implementation designed for solving the particular case of MC. In Section 6, we then explain how to extract an optimal solution of the max-cut problem from an optimal solution of its eigenvalue relaxation. Finally, a number of numerical results are presented in Section 7.
2 Preliminaries

As usual when dealing with optimization problems, we will use matrix notation to represent linear systems of equations in a compact and convenient way. Although we assume the notions of matrices and matrix operations to be familiar to the reader, Subsection 2.1 briefly describes the main notations that we are going to use in this paper. For the sake of convenience, we also recall in Subsection 2.2 a number of important results from linear algebra that are going to be useful in the following sections.

2.1 Notation

Let $m, n$ be integers and let us consider $\mathbb{R}^m$ and $\mathbb{R}^n$ as vector spaces over $\mathbb{R}$. Let also $M_{m,n}$ be the set of $m \times n$ real matrices. For a vector $x \in \mathbb{R}^n$ and a matrix $A \in M_{m,n}$, we denote the $j$-th component of $x$ by $x_j$ and the $(i, j)$-entry of $A$ by $a_{ij}$, where $1 \leq i \leq m$ and $1 \leq j \leq n$. We will use capital latin letters to denote matrices and small latin letters to denote vectors. Finally, we denote by $e := (1, 1, \ldots, 1)^T$ the vector of appropriate length whose coordinates are all ones and by $I$ the identity matrix of appropriate size.

If we interpret $M_{m,n}$ as a vector space in $\mathbb{R}^{n \cdot m}$, the natural inner product between two elements $A = (a_{ij})$ and $B = (b_{ij})$ in $M_{m,n}$ is the Frobenius inner product

$$\langle A, B \rangle = \text{tr}(B^T A) = \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} b_{ij},$$

where the trace $\text{tr}(\cdot)$ is the sum of the diagonal elements of a square matrix. Note that for vectors, the Frobenius inner product corresponds to the usual Euclidian inner product, i.e. $\langle x, y \rangle = y^T x$ for $x, y \in \mathbb{R}^n$. The trace is a linear operator, which implies that the Frobenius inner product is bilinear. The induced norm is the Frobenius norm,

$$\|A\|_F = \sqrt{\langle A, A \rangle}.$$

In this paper we shall mainly work with the set of symmetric matrices in $M_{n,n}$, denoted by $S_n$, which we naturally endow with the inner product of $M_{n,n}$. An arbitrary matrix $A \in S_n$ can always be diagonalized, which means there exist an orthonormal matrix $P \in M_{n,n}$ and a diagonal matrix $\Lambda \in S_n$ containing the eigenvalues of $A$ on its main diagonal such that $A = P \Lambda P^T$. All eigenvalues of a symmetric matrix $A$ are real and we denote them by $\lambda_i(A)$, $i = 1, \ldots, n$.

For our purposes we assume that the eigenvalues are sorted non-increasingly, namely $\lambda_{\max}(A) = \lambda_1(A) \geq \lambda_2(A) \geq \cdots \geq \lambda_n(A) = \lambda_{\min}(A)$. A matrix $A \in M_{n,n}$ is said to be positive semidefinite (respectively, positive definite) if $x^T A x \geq 0 \ \forall x \in \mathbb{R}^n$ (respectively, if $x^T A x > 0 \ \forall x \in \mathbb{R}^n \setminus \{0\}$). We write $A \succ 0$ when the matrix is symmetric positive semidefinite and $A \succ 0$ when it is symmetric positive definite.
2.2 Linear Algebra

For future reference and for the reader’s convenience, we state here a few basic results from linear algebra. We first recall the Rayleigh-Ritz Theorem, which provides the most common way to characterize the maximum eigenvalue of a symmetric matrix.

**Theorem 2.1 (Rayleigh-Ritz).** Let $A \in S_n$. Then

\[
\lambda_{\text{max}}(A) = \max_{\|v\|=1} v^T Av, \quad (2.2.1)
\]

\[
\lambda_{\text{min}}(A) = \min_{\|v\|=1} v^T Av. \quad (2.2.2)
\]

*Proof.* See Horn and Johnson [7].

In particular, the maximum in (2.2.1) is attained for eigenvectors to the maximum eigenvalue of $A$. Next, we recall the Singular Value Decomposition (SVD) of a matrix.

**Theorem 2.2 (SVD).** Let $A \in M_{m,n}$. Then there are orthogonal matrices $U \in M_{m,m}$ and $V \in M_{n,n}$ such that

\[
A = U\Sigma V^T, \quad \text{with} \quad \Sigma = \text{Diag}(\sigma_1, \ldots, \sigma_r, 0, \ldots, 0) \in M_{m,n}
\]

and $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r \geq 0$ for $r = \min\{m, n\}$.

*Proof.* See Horn and Johnson [7].

The uniquely determined scalars $\sigma_1, \ldots, \sigma_r$ in Theorem 2.2 are called the singular values of $A$. Let $A = U\Sigma V^T$ be an SVD of an arbitrary matrix $A \in M_{m,n}$. For any $k \leq \min\{m, n\}$, let $U_k \in M_{m,k}$ and $V_k \in M_{n,k}$ contain the first $k$ columns of $U$ and $V$, respectively. Then the matrix

\[
A_k := U_k\Sigma_k V_k^T, \quad \text{with} \quad \Sigma_k = \text{Diag}(\sigma_1, \ldots, \sigma_k)
\]

(2.2.3)

clearly has rank at most $k$. Moreover,

\[
\|A - A_k\|_F = \|\text{Diag}(\sigma_{k+1}, \ldots, \sigma_r)\|_F = \sqrt{\sigma_{k+1}^2 + \cdots + \sigma_r^2},
\]

which the following result shows to be minimal.

**Theorem 2.3 (Eckart-Young).** For a matrix $A \in M_{m,n}$, let $A_k$ be defined as in (2.2.3). Then

\[
\|A - A_k\|_F = \min \{\|A - B\|_F : B \in M_{m,n} \text{ has rank at most } k\}.
\]

*Proof.* See Horn and Johnson [7].
3 The Max-Cut Problem

Although it is one of the simplest graph partitioning problems to formulate, the max-cut problem (MC) is one of the most difficult combinatorial optimization problems to solve. Indeed, it belongs to the class of NP-complete problems, meaning that it is not yet known whether it can be solved with an algorithm running in polynomial time for general graphs (for planar graphs, it has been shown that MC is the dual to the route inspection problem, thus solvable in polynomial time [2]). It is interesting to note however that the inverse problem of MC, namely the min-cut problem, can be solved in polynomial time by means of network flow techniques [9]. Applications of MC appear in various fields such as VLSI circuit design [8] and statistical physics [1].

In this section we briefly introduce the standard and algebraic formulations of the max-cut problem.

3.1 Standard Formulation

The max-cut problem is defined in the framework of graph theory. Denote an undirected graph $G = (V, E)$ as a pair consisting of a set of vertices $V$ and a set of edges $E$. We assume that $G$ contains no loop nor multiple edges, that is there is no edge starting and ending at the same vertex and there can be at most one edge between two distinct vertices. We use $V = \{1, \ldots, n\}$ and $E = \{ij: i < j, i, j \in V\}$, i.e. an edge with endpoints $i$ and $j$ is denoted by $ij$. Let the map $w: E \rightarrow \mathbb{R}$ be a weight function defined on the set of edges.

In this paper, we shall only work with the general case of the complete edge-weighted graph $K_n$. Indeed, any graph can easily be modelled in this setting; it suffices to consider a non-existing edge as an edge with zero weight by setting $w_{ij} = 0$ if $ij \notin E$. For an unweighted graph we simply assign a unitary weight to every edge, namely $w_{ij} = 1$ for all $ij \in E$. The symmetric matrix $W = (w_{ij}) \in S_n$ is referred to as the weighted adjacency matrix of the graph. Since we assume $G$ to contain no loop, all the diagonal entries of $W$ are zero, hence $\text{tr}(W) = 0$.

The notion of cut is intimately related to that of partition of the vertex set. Indeed, a cut is defined as a set of edges such that the vertex set is partitioned in two disjoint subsets when those edges are removed. More precisely, for a subset $S \subset V$, the cut $\delta(S)$ is the set of edges that have one endpoint in $S$ and the other in $V \setminus S$, namely $\delta(S) = \{ij \in E: i \in S, j \in V \setminus S\}$.

The weight of a cut is the sum of the weights of its edges. The max-cut problem therefore consists in finding a cut of maximum weight. More formally, it can be written as

$$\text{(MC)} \quad \max_{S \subset V} \sum_{ij \in \delta(S)} w_{ij}. \quad (3.1.1)$$
Figure 3.1 shows a very simple example of a complete graph on 3 vertices corresponding to the adjacency matrix

\[
W = \begin{bmatrix}
0 & 1 & 4 \\
1 & 0 & 3 \\
4 & 3 & 0
\end{bmatrix}.
\]

In this example, we can see very easily that the maximum cut (represented by the red edges) has weight 7.

\[\begin{aligned}
&= 4 + 3 + 4 = 7.
\end{aligned}\]

3.2 Algebraic Formulation

In order to solve the max-cut problem, it will be more convenient to work with an algebraic formulation of (3.1.1). For a subset \(S \subset V\), let us introduce the associated cut vector \(x \in \{-1, 1\}^n\) defined by

\[
x_i = \begin{cases} 
1, & \text{if } i \in S \\
-1, & \text{if } i \in V \setminus S
\end{cases}
\]

for \(i = 1, \ldots, n\). Observe that for an edge \(ij \in E\), we have \(x_i = -x_j\) if \(ij \in \delta(S)\) and \(x_i = x_j\) if \(ij \notin \delta(S)\). Therefore the expression \(\frac{1}{2}(1 - x_ix_j)\) evaluates to 1 if the edge \(ij\) is in the cut \(\delta(S)\) and to 0 otherwise. \(\frac{1}{2}(1 - x_ix_j)\) is called the incidence vector associated to the cut. Since we have to count each edge in the cut only once, we can write

\[
\sum_{ij \in \delta(S)} w_{ij} = \sum_{i < j} w_{ij} \frac{1 - x_ix_j}{2}.
\]
Exploiting the symmetry of $W$ and the fact that $x_i x_i = 1$, this yields

$$\sum_{ij \in \delta(S)} w_{ij} = \frac{1}{4} \sum_{ij} w_{ij} (1 - x_i x_j)$$

$$= \frac{1}{4} \sum_{i=1}^{n} \left( \sum_{j=1}^{n} w_{ij} x_i x_j - \sum_{j=1}^{n} w_{ij} x_i x_j \right)$$

$$= \frac{1}{4} x^T (\text{Diag}(W e) - W) x,$$

where the linear operator $\text{Diag}(\cdot) : \mathbb{R}^n \to S_n$ is defined by

$$\text{Diag}(y) = \begin{bmatrix} y_1 & 0 & \cdots & 0 \\ 0 & y_2 & \vdots & \\ \vdots & \ddots & 0 \\ 0 & \cdots & 0 & y_n \end{bmatrix}$$

for all $y \in \mathbb{R}^n$. The matrix

$$L := \frac{1}{4} (\text{Diag}(W e) - W)$$

(3.2.2)

is called the Laplace matrix of the graph. Note that the symmetry of $W$ implies the symmetry of $L$. With this notation, we finally obtain an equivalent algebraic formulation of (3.1.1), namely

$$(\text{AMC}) \max_{x \in \{-1,1\}^n} x^T L x.$$
4 Semidefinite Programming

Semidefinite programming is a subfield of convex optimization that includes several common classes of optimization problems, such as for instance linear programming, quadratic programming, and semidefinite linear complementarity problems. In general, semidefinite programs (SDP) naturally arise from problems whose data is given by matrices. In addition, many relaxations of practical problems in operations research and combinatorial optimization can be formulated as SDP, which makes this relatively new field of optimization very interesting.

A close relation can be drawn between semidefinite programming and linear programming: instead of optimizing over the cone of the nonnegative orthant $x \geq 0$, in semidefinite programming the optimization is performed over the more general cone of symmetric positive semidefinite matrices $X \succeq 0$. However, unlike for linear programs, not every SDP satisfies strong duality.

In this section we first introduce our standard formulation of a primal semidefinite program and derive its dual. After a glimpse on the duality theory of semidefinite programming, we finally derive a semidefinite relaxation of the MAX-CUT problem.

4.1 The Standard Primal-Dual Pair of Semidefinite Programs

The standard formulation of a semidefinite program is

$$\text{(PSDP)} \quad \begin{array}{ll}
\text{max} & \langle C, X \rangle \\
\text{s.t.} & AX = b, \\
& X \succeq 0,
\end{array} \quad (4.1.1)
$$

where $C \in S_n$ is a given cost matrix and $X \in S_n$ is the matrix variable. The constraints are given by the vector $b \in \mathbb{R}^m$ and the linear operator $A : S_n \to \mathbb{R}^m$ defined by

$$AX = \begin{bmatrix} \langle A_1, X \rangle \\ \vdots \\ \langle A_m, X \rangle \end{bmatrix}, \quad (4.1.2)$$

where $A_i$ are given symmetric matrices for $i = 1, \ldots, m$.

As often in constrained optimization, it is possible to convert the primal problem to a dual form. Before deriving the dual of (PSDP), we need to determine the adjoint operator to $A$. By definition, it is the operator $A^T : \mathbb{R}^m \to S_n$ satisfying $\langle AX, y \rangle = \langle X, A^T y \rangle$ for all $X \in S_n$ and $y \in \mathbb{R}^m$. Exploiting the linearity of the inner product, we have

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

hence we obtain

$$A^T y = \sum_{i=1}^m y_i A_i. \quad (4.1.3)$$

In order to derive the dual of (PSDP) we follow a Lagrangian approach, that is we lift the $m$ primal equality constraints of (4.1.1) into the objective function by means of a Lagrangian multiplier $y \in \mathbb{R}^m$. By making this Lagrangian relaxation as tight as
possible, the primal problem reads \( \sup_{X \succeq 0} \inf_{y \in \mathbb{R}^m} [(C, X) + (b - AX, y)] \). The dual of (PSDP) is then constructed by swapping \( \inf \) and \( \sup \), leading to
\[
\sup_{X \succeq 0} \inf_{y \in \mathbb{R}^m} [(C, X) + (b - AX, y)] \leq \inf_{y \in \mathbb{R}^m} \sup_{X \succeq 0} [(b, y) + \langle X, C - AT y \rangle].
\] (4.1.4)

By construction, the value of the dual problem is an upper bound for the value of the primal problem. Conversely, the value of the primal problem is a lower bound for the value of the dual problem. An explicit justification of this fact will be given in (4.2.1). To ensure the finiteness of the infimum on the right hand side, the inner maximization over \( X \succeq 0 \) has to remain finite for some \( \hat{y} \in \mathbb{R}^m \). This requires \( \langle X, C - AT \hat{y} \rangle \leq 0 \) for all \( X \succeq 0 \), which by Fejer’s Trace Theorem is equivalent to \( AT \hat{y} - C \succeq 0 \) (see Helmberg [3], Corollary 1.2.7). In order to take this condition into account we introduce a slack variable \( Z := AT \hat{y} - C \in S_n \) and require it to be positive semidefinite. We finally obtain the standard formulation of the dual semidefinite program to (PSDP),
\[
\min \langle b, y \rangle \\
\text{s.t. } \quad AT y - Z = C, \quad (4.1.5)
\]
\[y \in \mathbb{R}^m, Z \succeq 0.\]

4.2 Duality Theory

A matrix \( X \in S_n \) is called a \textbf{primal feasible solution} if it satisfies the constraints of (PSDP), i.e. if \( AX = b \) and \( X \succeq 0 \). Similarly, a pair \((y, Z) \in \mathbb{R}^m \times S_n\) is called a \textbf{dual feasible solution} if it satisfies the constraints of (DSDP), i.e. \( AT y - Z = C \) and \( Z \succeq 0 \). In order to emphasize the importance of the inequality shown in (4.1.4), we consider the \textbf{duality gap} between a dual feasible solution \((y, Z)\) and a primal feasible solution \( X \)
\[
\langle b, y \rangle - \langle C, X \rangle = \langle AX, y \rangle - \langle AT y - Z, X \rangle = \langle Z, X \rangle \geq 0,
\] (4.2.1)
where the last inequality comes from the fact that both \( X \) and \( Z \) are positive semidefinite matrices. The property that the duality gap is always nonnegative is referred to as \textbf{weak duality}. In other words, the objective value of any primal feasible solution is at most the objective value of any dual feasible solution.

If the duality gap \( \langle Z, X \rangle \) is equal to zero, then the primal-dual triplet \((X, y, Z)\) is an optimal solution. However, unlike for linear programming, optimality no longer implies \( \langle Z, X \rangle = 0 \) in the case of semidefinite programming. Indeed, it may happen that primal and dual optimal objective values do not coincide. This is due to the fact that semidefinite sets of feasible solutions have a nonpolyhedral structure. Nevertheless, the following result of \textbf{strong duality} shows that the problem is well behaved if the primal or the dual feasible sets contain a \textit{positive definite} feasible point.

\textbf{Definition 4.1.} A point \( X \) is \textbf{strictly feasible} for (PSDP) if it is feasible for (PSDP) and satisfies \( X \succ 0 \). A pair \((y, Z)\) is \textbf{strictly feasible} for (DSDP) if it is feasible for (DSDP) and satisfies \( Z \succ 0 \).
**Theorem 4.2** (Strong Duality). Assume that there exists a strictly feasible solution for (DSDP) and let

\[ p^* := \sup \{ \langle C, X \rangle : AX = b, X \succ 0 \} \]
\[ d^* := \inf \{ \langle b, y \rangle : A^T y - Z = C, Z \succ 0 \} \]

Then \( p^* = d^* \) and if \( p^* \) is finite it is attained for some \( X \in \{ X \succ 0 : AX = b \} \).

**Proof.** See Helmberg [3], Theorem 2.2.5.

### 4.3 Semidefinite Relaxation of the Max-Cut Problem

In order to derive a semidefinite relaxation of the algebraic formulation (3.2.3) of the max-cut problem, observe that

\[ x^T L x = \sum_{i=1}^{n} \sum_{j=1}^{n} x_i L_{ij} x_j = \sum_{i=1}^{n} \sum_{j=1}^{n} L_{ij} (x_i x_j) = \langle L, xx^T \rangle \]

for all \( x \in \mathbb{R}^n \). By construction, the matrix \( xx^T \) has rank one and it is symmetric positive semidefinite. Moreover, for all \( x \in \{-1, 1\}^n \), its diagonal entries are equal to 1. Relaxing \( xx^T \) to an arbitrary matrix \( X \in S_n \) satisfying these three properties, we obtain

\[
\begin{align*}
\text{max} & \quad \langle L, X \rangle \\
\text{s.t.} & \quad \text{diag}(X) = e, \\
& \quad X \succ 0, \\
& \quad \text{rank}(X) = 1,
\end{align*}
\]

(4.3.1)

where \( \text{diag}(\cdot) : S_n \to \mathbb{R}^n \) is defined by

\[ \text{diag}(X) = (x_{11}, x_{22}, \ldots, x_{nn})^T \] (4.3.2)

for all \( X \in S_n \). The constraint \( \text{diag}(X) = e \) can be written in the form \( AX = b \) with \( b = e \) and \( A : S_n \to \mathbb{R}^n \) defined as in (4.1.2) with \( A_i \) being the \( n \times n \) matrix with a 1 at the \((i, i)\)-entry and zeros elsewhere. On the contrary, the constraint \( \text{rank}(X) = 1 \) cannot be expressed in the form \( AX = b \), hence (4.3.1) is not yet a semidefinite program. In fact, it is still equivalent to (3.2.3), because both problems optimize the same objective function over the same feasible set. Indeed, the constraints \( X \in S_n \) and \( \text{rank}(X) = 1 \) together imply that there exists a factorization \( x \in \mathbb{R}^n \) such that \( X = xx^T \). Moreover, the constraint \( \text{diag}(X) = e \) implies \( 1 = X_{ii} = x_i x_i \) for \( i = 1, \ldots, n \), which yields \( x \in \{-1, 1\}^n \).

As a consequence, we have to drop the rank one constraint in order to obtain a continuous feasible set and the primal semidefinite relaxation of (AMC) reads

\[
\begin{align*}
\text{max} & \quad \langle L, X \rangle \\
(P) & \quad \text{s.t.} \quad \text{diag}(X) = e, \\
& \quad X \succ 0.
\end{align*}
\]

(4.3.3)
In order to derive the corresponding dual problem, we first need to determine the dual operator of diag(·). In fact, it is the linear operator Diag(·) defined in (3.2.1). Indeed, for all \( X \in S_n \) and \( y \in \mathbb{R}^n \) we have

\[
\langle \text{diag}(X), y \rangle = \sum_{i=1}^{n} x_{ii} y_i = \sum_{i=1}^{n} x_{ii} (\text{Diag}(y))_{ii} = \sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij} (\text{Diag}(y))_{ij} = \langle X, \text{Diag}(y) \rangle.
\]

Therefore, according to (4.1.5), the dual problem to (P) is

\[
\min \langle e, y \rangle \quad \text{s.t.} \quad \text{Diag}(y) - Z = L, \quad y \in \mathbb{R}^n, Z \succeq 0.
\]

### 4.4 Solving the Semidefinite Relaxation with SeDuMi

At this point, provided that the dimension \( n \) of the problem is not too large (see section 5), we could already solve the semidefinite relaxations (P) and (D) in order to obtain an approximation of the optimal solution to the original max-cut problem (3.2.3). A very convenient and effective way to do so is given by the MATLAB toolbox SeDuMi [11] (stands for Self-Dual-Minimization), which allows to solve optimization problems over symmetric cones.

SeDuMi implements the self-dual embedding technique proposed by Ye, Todd and Mizuno [12]. In particular, SeDuMi offers the advantage to state a problem in MATLAB in a very similar way as one would formulate it in theory. The problem has to be stated between two tags between which the SeDuMi routines are called.

Assuming that the Laplace matrix \( L \) of a graph on \( n \) vertices is given, we can solve the primal semidefinite relaxation (P) of the max-cut problem with the MATLAB code:

```matlab
1 cvx_begin sdp
2 variable X(n,n) symmetric
3 maximize(sum(sum(L.*X)))
4 diag(X) == e
5 X >= 0
6 cvx_end
```

In the same way, the dual semidefinite relaxation (D) of the max-cut problem can be solved with the MATLAB code:

```matlab
1 cvx_begin sdp
2 variable y(n)
3 variable Z(n,n) symmetric
4 minimize(sum(y))
5 Z == diag(y) - L
6 Z >= 0
7 cvx_end
```

In section 7 we will use the solutions obtained by SeDuMi as a way to verify the accuracy of the solutions obtained by our MATLAB implementation of the Spectral Bundle Method.
5 Eigenvalue Optimization

Optimization problems involving eigenvalues arise in several mathematical disciplines. When optimization is performed over affine sets of matrices, eigenvalue optimization is closely related to semidefinite programming. This is due to the fact that a symmetric matrix $X \in S_n$ is positive semidefinite if and only if its minimal eigenvalue is nonnegative, namely $\lambda_{\text{min}}(X) \geq 0$. In this section we make this relation more precise and give a standard formulation of an eigenvalue optimization problem. Then we derive an eigenvalue relaxation of the MAX-CUT problem. We also present a complete description of the Spectral Bundle Method, which gives rise to an iterative algorithm for solving eigenvalue optimization problems. Finally, we implement this method in MATLAB in order to solve the particular case of the MAX-CUT problem.

5.1 Standard Formulation

Let us consider again the dual semidefinite program in standard form

$$\min \langle b, y \rangle$$

\[ \text{s.t.} \quad Z = A^T y - C, \]

\[ y \in \mathbb{R}^m, Z \succeq 0, \]

that we introduced in Section 4. In order to reformulate (DSDP) as an eigenvalue optimization problem, we use the fact that $Z \succeq 0 \iff 0 \leq \lambda_{\text{min}}(Z) = -\lambda_{\text{max}}(-Z) \iff \lambda_{\text{max}}(-Z) \leq 0$

and lift this constraint in the objective function by means of a Lagrangian multiplier $a \geq 0$. This yields our standard formulation of an eigenvalue optimization problem

$$(EOP) \quad \min_{y \in \mathbb{R}^m} [a\lambda_{\text{max}}(C - A^T y) + b^T y]. \quad (5.1.1)$$

For the sake of convenience we will denote the objective function by

$$f(y) := a\lambda_{\text{max}}(C - A^T y) + b^T y \quad (5.1.2)$$

in the sequel of this paper. Under the assumption that the $n \times n$ identity matrix belongs to the range of the operator $A^T$, problem (EOP) is equivalent to (DSDP). This important result is captured in the following proposition (see Helmberg [3], Proposition 5.1.1).

**Proposition 5.1.** Assume there exists $\bar{y} \in \mathbb{R}^m$ such that $I = A^T \bar{y}$. Then (DSDP) is equivalent to (EOP) for $a = \max\{0, b^T \bar{y}\}$.

**Proof.** Let $\bar{y} \in \mathbb{R}^m$ be such that $I = A^T \bar{y}$ and for $y \in \mathbb{R}^m$ consider the half ray $R(y) := \{y + \lambda \bar{y} : \lambda \geq \lambda_{\text{max}}(C - A^T y)\}$. Note that by construction, $R(y)$ is feasible for (DSDP) for all $y \in \mathbb{R}^m$. Indeed, due to the condition on $\lambda$ we have

$$A^T (y + \lambda \bar{y}) - C = \lambda A^T \bar{y} + A^T y - C$$

$$= \lambda I - (C - A^T y) \succeq 0.$$  

In particular, this implies that (DSDP) has strictly feasible solutions (just choose $\lambda > \lambda_{\text{max}}(C - A^T y)$ above). We now consider the two cases $b^T \bar{y} < 0$ and $b^T \bar{y} \geq 0$:
Case 1: Assume $b^T \bar{y} < 0$. Then $a = \max\{0, b^T \bar{y}\} = 0$ and the objective values of both (DSDP) and (EOP) tend to minus infinity along any ray $y + \lambda \bar{y}$ when $\lambda \to \infty$.

Case 2: Assume $b^T \bar{y} \geq 0$. Then $a = \max\{0, b^T \bar{y}\} = b^T \bar{y} \geq 0$. We first show that for any feasible solution $y \in \mathbb{R}^m$ of (EOP) there exists a feasible solution $\hat{y}$ of (DSDP) with the same objective value. Let $y \in \mathbb{R}^m$ be a given feasible solution of (EOP). Note that problem (EOP) is constant along directions $\lambda \bar{y}$ for $\lambda \in \mathbb{R}$, because

$$f(y + \lambda \bar{y}) = a \lambda \max(C - A^T(y + \lambda \bar{y})) + b^T(y + \lambda \bar{y})$$

$$= a \lambda \max(C - A^T y - \lambda I) + b^T y + \lambda a$$

$$= a \lambda \max(C - A^T y) + b^T y$$

$$= f(y).$$

In particular, the choice $\hat{\lambda} := \lambda \max(C - A^T y)$ ensures that the term $\lambda \max(C - A^T(y + \lambda \bar{y}))$ vanishes, hence $\hat{y} := y + \hat{\lambda} \bar{y} \in R(y)$ is a feasible solution of (DSDP) satisfying $b^T \hat{y} = f(\hat{y}) = f(y)$.

Conversely, we show that for any feasible solution $y \in \mathbb{R}^m$ of (DSDP) there exists a feasible solution $\hat{y}$ of (EOP) with at most the same objective value. Let $y \in \mathbb{R}^m$ be a given feasible solution of (DSDP). Consider the point $\hat{y} := y + \lambda \max(C - A^T y) \bar{y} \in R(y)$. Since $y$ is feasible for (DSDP) we have

$$Z = A^T y - C \succeq 0 \quad \iff \quad \lambda \max(C - A^T y) \leq 0,$$

therefore we obtain

$$f(\hat{y}) = b^T \hat{y} = b^T y + \lambda \max(C - A^T y) a \leq b^T y.$$

Because (EOP) is a minimization problem, this suffices to prove the equivalence of (DSDP) and (EOP).

**Remark 5.2.** If (PSDP) is feasible, all its feasible solutions satisfy $\text{tr}(X) = a$, because

$$0 = \langle AX - b, \bar{y}\rangle = \langle X, A^T \bar{y} \rangle - b^T \bar{y} = \langle X, I \rangle - a = \text{tr}(X) - a.$$

Moreover, since (DSDP) has strictly feasible solutions, the Strong Duality Theorem 4.2 implies that the primal optimum is attained and is equal to the infimum of (DSDP).

### 5.2 Eigenvalue Relaxation of the Max-Cut Problem

In the particular case of the MAX-CUT problem, the dual semidefinite relaxation is

$$\begin{align*}
\min & \quad e^T y \\
\text{s.t.} & \quad Z = \text{Diag}(y) - L, \\
& \quad y \in \mathbb{R}^n, Z \succeq 0,
\end{align*}$$

(D)
where the adjoint $A^T$ of the linear operator $A$ is $\text{Diag}(\cdot) : \mathbb{R}^n \to S_n$. Clearly we have $I = \text{Diag}(e)$, which yields $\bar{y} = e$. Therefore we have $a = \max\{0, e^T e\} = n > 0$ and the eigenvalue relaxation of the MAX-CUT problem reads

$$
\text{(E)} \quad \min_{y \in \mathbb{R}^m} [n \lambda_{\max}(L - \text{Diag}(y)) + e^T y].
$$

(5.2.1)

In view of Proposition 5.1, we have the equivalence between problems (D) and (E).

### 5.3 Analysis of the Objective Function

In this subsection we go back to the general case and analyse the cost function of problem (EOP), namely $f(y) = a \lambda_{\max}(C - A^Ty) + b^Ty$. As the case $a = 0$ is of little interest, we assume that $a > 0$ in the rest of this section. To begin with, we investigate the convex structure of the maximum eigenvalue function $\lambda_{\max}(\cdot)$ as well as its subdifferential. We first use the Rayleigh-Ritz Theorem 2.1, which allows us to characterize the maximum eigenvalue of a symmetric matrix as

$$
\lambda_{\max}(X) = \max_{\|v\|=1} v^TXv.
$$

(5.3.1)

The next result will be very helpful in order to reformulate (5.3.1).

**Proposition 5.3.** The set $W = \{W \in S_n : W \succeq 0, \text{tr}(W) = 1\}$ is the convex hull of the set $V = \{vv^T : v \in \mathbb{R}^n, \|v\| = 1\}$.

**Proof.** It is sufficient to show that any $W \in W$ can be written as a convex combination of elements in $V$. Let $W \in W$ be given. Since $W$ is symmetric, a Singular Value Decomposition (see Theorem 2.2) has the form

$$
W = V \Sigma V^T,
$$

where $V \in M_{n,n}$ is an orthogonal matrix and $\Sigma = \text{Diag}(\sigma_1, \ldots, \sigma_n)$. Denoting the $i$-th column of $V$ by $v_i$, we obtain

$$
W = \sum_{i=1}^n \sigma_i v_i v_i^T.
$$

(5.3.2)

Since $V$ is orthogonal we have $\|v_i\|^2 = v_i^Tv_i = 1$, hence $v_i v_i^T \in V$ for $i = 1, \ldots, n$. Moreover, the singular values satisfy $\sigma_i \geq 0$ for $i = 1, \ldots, n$ and we have

$$
\sum_{i=1}^n \sigma_i = \text{tr}(\Sigma) = \text{tr}(W) = 1,
$$

where we used the fact that the trace is invariant under similarity transformations. This proves that (5.3.2) is a convex combination. $\square$

Using the fact that $v^TXv = \langle X, vv^T \rangle$ for all $v \in \mathbb{R}^n$, the linearity of the Frobenius scalar product and Proposition 5.3, maximizing over $V$ in (5.3.1) amounts to maximizing over $W$ and thus we can reformulate the maximum eigenvalue function as a semidefinite program,

$$
\lambda_{\max}(X) = \max_{W \in W} \langle X, W \rangle.
$$

(5.3.3)
As the maximum over a family of linear functions, \( \lambda_{\text{max}}(\cdot) \) is convex. This brings us to the notion of subgradient. The subgradients of \( \lambda_{\text{max}}(\cdot) \) at \( X \) are the linear forms \( W_S \) that satisfy the subgradient inequality \( \lambda_{\text{max}}(Y) \geq \lambda_{\text{max}}(X) + \langle Y - X, W_S \rangle \) for all \( Y \in S_n \). The subdifferential of \( \lambda_{\text{max}}(\cdot) \) at \( X \) is the set of all subgradients of \( \lambda_{\text{max}}(\cdot) \) at \( X \) and is denoted by \( \partial \lambda_{\text{max}}(X) \). In particular, we have the characterization

\[
\partial \lambda_{\text{max}}(X) = \text{Argmax}_{W \in W} \langle X, W \rangle = \{ PV P^T : \text{tr}(V) = 1, V \succeq 0 \},
\]  

(5.3.4)

where the columns of \( P \) form an orthonormal basis of the eigenspace to the maximum eigenvalue of \( X \) (see Helmberg [3], p.74).

**Remark 5.4.** Due to Proposition 5.3, the subdifferential in (5.3.4) may be viewed as the convex hull of the dyadic products of the normalized eigenvectors to the maximum eigenvalue of \( X \). In particular, any eigenvector \( v \in \mathbb{R}^n \) to \( \lambda_{\text{max}}(X) \) gives rise to a subgradient of \( \lambda_{\text{max}}(\cdot) \) at \( X \) of the form \( W_S = vv^T \).

Let us now consider the function \( \alpha \lambda_{\text{max}}(X) = \max_{W \in W} \langle X, aW \rangle \). Using the linearity of the trace operator, we have \( \text{tr}(aW) = a \text{tr}(W) = a \) for all \( W \in W \). Moreover, \( aW \succeq 0 \iff W \succeq 0 \) (recall that we assumed \( a > 0 \)), so that if we set

\[
\mathcal{W}_a := \{ W \in S_n : W \succeq 0, \text{tr}(W) = a \}
\]

(5.3.5)

we can write

\[
\alpha \lambda_{\text{max}}(X) = \max_{W \in \mathcal{W}_a} \langle X, W \rangle,
\]

whose subdifferential is

\[
\partial \alpha \lambda_{\text{max}}(X) = \{ PV P^T : \text{tr}(V) = a, V \succeq 0 \}.
\]

Finally, this allows us to reformulate the objective function \( f \) of (5.1.2) as

\[
f(y) = \max_{W \in \mathcal{W}_a} [(C, W) + \langle b - AW, y \rangle].
\]

(5.3.6)

The convexity of \( \lambda_{\text{max}} \) implies that of \( f \). As a consequence, the subdifferential of \( f \) at \( y \) is

\[
\partial f(y) = \left\{ b - AW : W \in \partial \alpha \lambda_{\text{max}}(C - A^T y) \right\}.
\]

(5.3.7)
5.4 The Spectral Bundle Method

The Spectral Bundle Method is a specialized subgradient method for solving eigenvalue optimization problems of the form (EOP). It was developed by C. Helmberg [3] as an alternative for solving the class of large constrained semidefinite programs that can be cast as eigenvalue optimization problems. Indeed, although primal-dual interior point methods work well on reasonably defined semidefinite programs in theory, their applicability is limited to problems with about 7000 constraints on a workstation equipped with the current technology [3]. This is due to the fact that the system matrix arising in the computation of the step direction is in general dense and positive definite, which often makes the cost of a single iteration too important. The Spectral Bundle Method allows to solve high dimensional problems of the form (EOP) (in particular, problems arising from semidefinite programs with a large number of constraints), but has the drawback of having a poor convergence rate: it is only a first order method.

Before giving a more precise mathematical description of the Spectral Bundle Method and its algorithm, we begin with a verbal description of the general principles of the method.

5.4.1 Verbal Description

Due to the maximum eigenvalue function \( \lambda_{\text{max}}(\cdot) \), the objective function \( f \) of (5.1.2) that we want to minimize is a nonsmooth convex function (see Subsection 5.3). Therefore, the Spectral Bundle Method employs nonsmooth convex optimization techniques to solve (EOP) and requires the computation of a subgradient of \( f \) at a given point \( y \in \mathbb{R}^m \). Together with the function value \( f(y) \), a subgradient gives rise to a linear minorant of \( f \) that touches \( f \) at \( y \). This linear function minorizing \( f \) is usually referred to as a supporting hyperplane or a cutting surface model of \( f \) at \( y \).

Rather than minimizing \( f \) directly, subgradient methods work with a sequence of cutting surface models of \( f \) and therefore an oracle returning \( f(y) \) and providing some subgradient \( s \in \partial f(y) \) for a given point \( y \) is the only necessary information for such methods. In view of Remark 5.4, subgradients to the maximum eigenvalue function at \( X \) are completely determined by the eigenspace to the maximum eigenvalue of \( X \). Therefore, as long as the matrix structure enables quick matrix-vector multiplications, maximal eigenvalues and eigenvectors of large scaled matrices can be efficiently determined by iterative methods such as for instance the Lanczos method (see [10]). In this case, function value and subgradient can be provided by the oracle fairly quickly.

In particular, the Spectral Bundle Method is a specialized subgradient method which produces a sequence of semidefinite cutting surface models of \( f \) by accumulating and updating subgradient information in a matrix called the bundle. This bundle matrix determines the size of a convex quadratic semidefinite subproblem that has to be solved at each iteration, hence the importance of keeping the size of the bundle matrix relatively small. Subgradient information that is removed from the bundle matrix is accumulated in an other matrix called the aggregate.
5.4.2 Mathematical Description

We will now describe one iteration of the Spectral Bundle Method in more detail. We also propose a MATLAB implementation of the method for the particular case of the max-cut problem. Recall that the goal is to minimize the cost function

\[ f(y) = a \lambda_{\text{max}}(C - A^T y) + b^T y \]

over \( y \in \mathbb{R}^m \) with \( a > 0 \). Assume that we are at iteration \( k \) and denote the \( k \)-th iterate by \( y^k \). The current center of stability \( \hat{y}^k \) is the starting point or the last successful iterate.

The Oracle

As a specialized subgradient method, the Spectral Bundle Method requires an oracle able to provide the objective value \( f(y^k) \) and a subgradient \( s^k \in \partial f(y^k) \) for some given iterate \( y^k \in \mathbb{R}^m \). Due to the structure of \( \partial f(y^k) \) (cf. (5.1.2)), the oracle is therefore assumed to deliver the maximum eigenvalue of a given matrix \( C - A^T y^k \) and a subgradient

\[ W^k_S \in \text{Argmax}_{W \in \mathcal{W}_a} (C - A^T y^k, W). \]

Remark 5.4 implies that we can take \( W^k_S \) of the form \( W^k_S = av^k(v^k)^T \), where \( v^k \) is an eigenvector to the maximum eigenvalue of \( C - A^T y^k \). In practise, the maximum eigenvalue and a corresponding eigenvector are usually computed by the Lanczos method, which is the best approach for computing a few extremal eigenvalues of a matrix whose product with a vector can be carried out quickly. We refer to Saad [10] for a full description of the Lanczos process.

Our implementation of the algorithm for the max-cut problem uses the Lanczos method with complete orthogonalization \( \text{lanczos\_orth} \) A.1.1 as soon as the dimension \( m \) of the problem is greater than 100. For small dimensions we simply use the MATLAB function \( \text{eig} \) in order to compute the eigenvectors and eigenvalues of a given symmetric matrix. The oracle is implemented in the MATLAB function \( \text{oracle} \) A.1.2.

The Cutting Surface Model

In each iteration of the Spectral Bundle Method, a minorizing cutting surface model of \( f \) at the current center of stability \( \hat{y}^k \) has to be computed. Characterization (5.3.6) shows that any fixed \( W \in \mathcal{W}_a \) gives rise to a linear function

\[ f_W(y) := \langle C, W \rangle + \langle b - AW, y \rangle \]

minorizing \( f \). Therefore, a subset \( \hat{W} \subset \mathcal{W}_a \) yields a convex minorant \( f_{\hat{W}} \) of \( f \) (i.e., a cutting surface model) as the pointwise maximum over those linear functions, namely

\[ f_{\hat{W}}(y) := \max_{W \in \hat{W}} f_W(y) \leq \max_{W \in \mathcal{W}_a} f_W(y) = f(y) \quad \text{for all} \quad y \in \mathbb{R}^m. \quad (5.4.1) \]

In order to exploit the structure of (EOP), it is very important to choose an appropriate cutting surface model for \( f \) (see Remark 5.7). Following Helmberg [3], the particular choice for iteration \( k \) is

\[ \hat{W}^k = \left\{ P_k V P_k^T + \alpha W_k : \text{tr}(V) + \alpha = a, V \succ 0, \alpha \geq 0 \right\}, \quad (5.4.2) \]
where $P_k$ is an orthonormal matrix of size $n \times r_k$ and $\overline{W}_k$ is an $n \times n$ positive semidefinite matrix of trace 1 (i.e., $\overline{W}_k \in \mathcal{W}$). We refer to $P_k$ as the bundle and to $\overline{W}_k$ as the aggregate, while the number of columns $r_k$ of $P_k$ is the size of the bundle. Both $P_k$ and $\overline{W}_k$ maintain relevant subgradient information and are updated at the end of each iteration. The cutting surface model corresponding to this subset is therefore $f_{\overline{W}_k}$ (cf. (5.4.1)), for which a characterization is given in the following proposition.

**Proposition 5.5.** For $\overline{W}_k$ of (5.4.2) and $f_{\overline{W}_k}$ defined as in (5.4.1),

$$f_{\overline{W}_k}(y) = \max \left\{ \lambda_{\max}(P_k^T (C - A^T y)P_k), (C - A^T y, \overline{W}_k) \right\} + b^T y \leq f(y).$$

**(5.4.3)**

**Proof.** See Helmberg [3], Proposition 5.2.1.

This proposition shows that the maximum eigenvalue of $P_k^T (C - A^T y)P_k$, and hence the value of the cutting surface model, can be efficiently determined provided that the bundle size $r_k$ remains small. Moreover, it helps to make a clever choice as how to construct the bundle matrix $P_k$. Indeed, since $f_{\overline{W}_k}$ is a minorant of $f$, it is best to have as large values of $f_{\overline{W}_k}$ as possible in the vicinity of the current stability center $\hat{y}_k$ in order to obtain a good approximation of $f$ around $\hat{y}_k$. This is the case if the columns of $P_k$ span the eigenspaces of the largest eigenvalues of $C - A^T \hat{y}_k$.

The Augmented Model

Since $f_{\overline{W}_k}$ is a linear approximation of $f$, it can be expected to be of reasonable quality only locally. Therefore the next iterate $y^{k+1}$ is sought in a neighborhood of $\hat{y}_k$ and determined by minimizing the augmented model

$$f^k(y) := f_{\overline{W}_k}(y) + \frac{u}{2} \|y - \hat{y}_k\|^2, \quad u > 0,$$

**(5.4.4)**

which is the sum of the cutting surface model $f_{\overline{W}_k}$ and a positive quadratic term penalizing the distance from $\hat{y}_k$. More precisely, $y^{k+1}$ is the minimizer of the problem

$$\min_{y \in \mathbb{R}^m} \max_{W \in \mathcal{W}_k} \left[ \langle C, W \rangle + \langle b - AW, y \rangle + \frac{u}{2} \|y - \hat{y}_k\|^2 \right].$$

**(5.4.5)**

In particular, the quadratic term $\frac{u}{2} \|y - \hat{y}_k\|^2$ ensures the minimum to be unique and finite. The parameter $u > 0$ allows to control (to some extent) the distance between $y^{k+1}$ and $\hat{y}_k$. It is intended to keep $y^{k+1}$ in the region where $f_{\overline{W}_k}$ should be close to $f$. For the sake of convenience, we introduce the augmented lagrangian as

$$L^k(y, W) := \langle C, W \rangle + \langle b - AW, y \rangle + \frac{u}{2} \|y - \hat{y}_k\|^2.$$

**(5.4.6)**

Rather than solving (5.4.5) we solve its dual, namely

$$\max_{W \in \mathcal{W}_k} \min_{y \in \mathbb{R}^m} L^k(y, W).$$

**(5.4.7)**
The inner minimization \( \min_{y \in \mathbb{R}^m} L^k(y, W) \) being an unconstrained convex quadratic problem, we can solve it explicitly for any fixed \( W \in \tilde{W}^k \). From
\[
0 = \nabla_y L^k(y, W) = b - AW + \frac{u}{2} (2y - 2\hat{y}^k)
\]
we obtain \( y^k_{\text{min}}(W) := \hat{y}^k + \frac{1}{u}(AW - b) \), so that
\[
L^k(y^k_{\text{min}}(W), W) = \langle C, W \rangle + \langle b - AW, \hat{y}^k + \frac{1}{u}(AW - b) \rangle + \frac{u}{2} \| \hat{y}^k + \frac{1}{u}(AW - b) - \hat{y}^k \|^2
\]
\[
= \langle C, W \rangle + \langle b - AW, \hat{y}^k \rangle - \frac{1}{u} \langle AW - b, AW - b \rangle + \frac{1}{2u} \| AW - b \|^2
\]
\[
= \langle C - A^T \hat{y}^k, W \rangle + b^T \hat{y}^k - \frac{1}{2u} \| AW - b \|^2.
\]
Due to the definition (5.4.2) of \( \tilde{W}^k \), maximizing \( L^k(y^k_{\text{min}}(W), W) \) over \( \tilde{W}^k \) amounts to solve the following quadratic semidefinite programming problem in \( V \) and \( \alpha \)
\[
\min \quad \frac{1}{2u} \| AW - b \|^2 - \langle C - A^T \hat{y}^k, W \rangle - b^T \hat{y}^k
\]
\[
\text{s.t.} \quad W = P_k V P_k^T + \alpha \tilde{W}_k, \quad \text{tr}(V) + \alpha = a, \quad V \succeq 0, \alpha \geq 0.
\]
The dimension of problem (Q) depends on the bundle size \( r_k \), which again emphasizes the importance of keeping the bundle relatively small throughout the algorithm. The following result shows that strong duality holds for the duals (5.4.5), (5.4.7).

**Proposition 5.6.** Let \( L^k \) be the augmented lagrangian as defined in (5.4.6). Then
\[
\min_{y \in \mathbb{R}^m} \max_{W \in \tilde{W}^k} L^k(y, W) = L^k(y^{k+1}, W^{k+1}) = \max_{W \in \tilde{W}^k} \min_{y \in \mathbb{R}^m} L^k(y, W) \quad (5.4.9)
\]
with \( y^{k+1} = y^k_{\text{min}}(W^{k+1}) \) unique and \( W^{k+1} \) an optimal solution of (Q).

**Proof.** See Helmberg [3], Lemma 5.2.2. \( \square \)

**Remark 5.7.** Of course, (5.4.2) is not the unique way to construct the subset \( \tilde{W}^k \subset \mathcal{W}_a \). Depending on the matrix structure of problem (EOP), various other choices of sets \( \tilde{W}^k \) may turn out useful in particular applications. The main design criterion for a model (i.e. for the choice of a set \( \tilde{W}^k \)) is the efficiency in solving the corresponding quadratic semidefinite subproblem. Determining a good model requires to find a compromise between the quality of the model, which strongly influences the number of iterations needed to achieve the desired precision, and the cost of evaluating \( f \) itself.

The structure (5.4.2) of the subset \( \tilde{W}^k \subset \mathcal{W}_a \) yields the convex quadratic semidefinite subproblem (Q) that can be solved fairly efficiently by means of interior point methods (provided that the bundle size is not too large). We refer to Helmberg [3], Section 5.5, for a description of such a method that exploits the particular structure of the problem. See also Helmberg and Rendl [6]; Helmb erg and Kiwiel [5] for a complete description of the interior point algorithm.

As this paper is only a humble semester project, our implementation of the algorithm for the MAX-CUT problem uses the MATLAB toolbox SeDuMi [11] for solving the quadratic subproblem (Q). This step is implemented in the MATLAB function trial_point A.1.3.
The Descent Test

The candidate \( y^{k+1} \) is then accepted as the new stability center only if the progress of its function value \( f(y^{k+1}) \) with respect to \( f(y^k) \) is satisfactory compared to the decrease predicted by the model value \( f_{W^{k+1}}(y^{k+1}) \), i.e. if

\[
f(\hat{y}^k) - f(y^{k+1}) \geq \kappa \left[ f(\hat{y}^k) - f_{W^{k+1}}(y^{k+1}) \right]
\]

(5.4.10)

for some parameter \( \kappa \in (0, 1) \). If this condition is satisfied this yields a descent step and the current center of stability is updated as \( \hat{y}^{k+1} := y^{k+1} \), otherwise it is called a null step and the current stability center remains unchanged, namely \( \hat{y}^{k+1} := \hat{y}^k \).

The Model Update

After a descent or a null step, the cutting surface model is updated. More precisely, a new subset \( \mathcal{W}^{k+1} \subset \mathcal{W}_k \) of the form (5.4.2) is determined by updating the bundle and the aggregate. Information from the new subgradient \( \mathcal{W}^{k+1}_S = \alpha v^{k+1} (v^{k+1})^T \), where \( v^{k+1} \) is an eigenvector to the maximum eigenvalue of \( C - A^T y^{k+1} \), is added to the model by adding the new eigenvector \( v^{k+1} \) to the old bundle matrix \( P_k \), hence increasing the bundle size. In order to still be able to solve the quadratic subproblem (Q) efficiently, some of the columns of \( P_k \) have to be removed in this process and their contribution to the solution is accumulated in the new aggregate matrix. The important information in \( \mathcal{W}^k \) that has to be maintained in \( \mathcal{W}^{k+1} \) is contained in the matrix \( W^{k+1} \). Indeed, as \( (y^{k+1}, W^{k+1}) \) is a saddle point of \( L^k \) (see Proposition 5.6), \( W^{k+1} \) ensures that after a null step the value of the augmented model cannot decrease. This forces a sequence of null steps to eventually produce a descent step over time. In fact, it has been shown (see Helmberg [3], Section 5.3) that convergence of the Spectral Bundle Method is guaranteed provided that

\[
W^{k+1}, W^{k+1}_S \in \mathcal{W}^{k+1} \quad \text{with} \quad \mathcal{W}^{k+1} \text{ of the form (5.4.2)}.
\]

(5.4.11)

The minimal choice within this framework is \( \mathcal{W}^{k+1} = W^{k+1} \) and \( P_{k+1} = v^{k+1} \). However, although \( r_{k+1} = 1 \) would suffice in theory, in practice it is well-advanced to preserve also the most important part of the subspace spanned by the columns of \( P_k \). This allows to accumulate the relevant eigenspace information without recomputing the whole spectrum in each iteration. The matrix \( W^{k+1} \) helps to determine which subspace has been important in the last computation. Denoting by \( V_s \) and \( \alpha_s \) the optimal solutions of (Q) that gave rise to \( W^{k+1} \), consider \( QAQ^T = V_s \) an eigenvalue decomposition of \( V_s \), where \( Q^T Q = I \) and \( \Lambda = \text{diag}(\lambda_1, \ldots, \lambda_{r_k}) \) with \( \lambda_1 \geq \cdots \geq \lambda_{r_k} \geq 0 \). Then we can write

\[
W^{k+1} = P_k V_s P_k^T + \alpha_s \bar{W}_k = (P_k Q) \Lambda (P_k Q)^T + \alpha_s \bar{W}_k,
\]

(5.4.12)

which shows that more information about \( W^{k+1} \) is carried by the first columns of \( P_k Q \) than by the last ones. Assuming therefore that the first columns of \( P_k Q \) are more important for the remaining part of the optimization process, we split \( Q \) into two parts, \( Q = [Q_1, Q_2] \), where \( Q_1 \) carries the eigenvectors corresponding to the largest

\footnote{obtained by the oracle}
eigenvalues of $V$. The matrix $\Lambda$ is analogously split into two smaller diagonal matrices $\Lambda_1$ and $\Lambda_2$. The new bundle matrix $P_{k+1}$ will then contain an orthonormal basis of the space spanned by the columns of $P_kQ_1$ and $v^{k+1}$, namely

$$P_{k+1} = \text{orth}[P_kQ_1, v^{k+1}].$$  \hfill (5.4.13)$$

The remaining columns $P_kQ_2$ are finally incorporated in the new aggregate matrix as

$$\overline{W}_{k+1} = (P_kQ_2)\Lambda_2(P_kQ_2)^T + \alpha_s \overline{W}_k,$$  \hfill (5.4.14)$$

where the scaling factor $1/\left(\text{tr } \Lambda_2 + \alpha_s\right)$ ensures that $\overline{W}_{k+1}$ has a trace equal to 1.

The following result shows that updating the bundle and the aggregate in this way maintains the structure (5.4.2) of the set $\overline{W}^k$.

**Proposition 5.8.** For $W_{k+1}^+ = a_s v^{k+1} (v^{k+1})^T \in \mathcal{W}_a$, update formulas (5.4.13) and (5.4.14) ensure that $P_{k+1}$ is orthonormal, $\overline{W}_{k+1} \in \mathcal{W}$, and that condition (5.4.11) is satisfied for $\overline{W}^{k+1}$ constructed as in (5.4.2).

**Proof.** See Helmberg [3], Proposition 5.2.3.

Our implementation of the algorithm for the max-cut problem adds 7 new eigenvectors to the bundle at each iteration and keeps the bundle size bounded using a parameter $r_{\text{max}} \in \mathbb{N}$ as an upper bound. The model update process is implemented in the MATLAB function `model_updating` A.1.4.

**The Stopping Criterion**

It remains to determine a reasonable stopping criterion. Of course, we would like the algorithm to stop when the objective value $f(\hat{y}^k)$ at the current stability center is close enough to the optimal value $\min_{y \in \mathbb{R}^m} f(y)$. However, a lower bound for the latter value is unfortunately not available. Indeed, although we know that the cutting surface model $f_{\overline{W}^k}$ is a minorant for $f$, we do not know its minimizer. Instead, we do know the minimizer $y^{k+1}$ of the augmented model. Considering the quadratic term of the augmented model as a kind of trust region constraint for the cutting surface model, $y^{k+1}$ may be viewed as the minimizer of $f_{\overline{W}^k}$ over a ball around $\hat{y}^k$. Provided that the weight $u$ remains reasonably small, the value $f_{\overline{W}^k}(y^{k+1}) = f_{W^{k+1}}(y^{k+1})$ provides a lower bound for $f$ over a ball of reasonable size. Therefore, if the gap between $f(\hat{y}^k)$ and $f_{W^{k+1}}(y^{k+1})$ is small, namely if

$$f(\hat{y}^k) - f_{W^{k+1}}(y^{k+1}) < \epsilon(|f(\hat{y}^k)| + 1),$$  \hfill (5.4.15)$$

no satisfactory progress can be expected within the trust region and we stop the algorithm.
5.4.3 The Algorithm

Here we state the algorithm in pseudo-code. For the particular case of the max-
cut problem, the whole algorithm is implemented in the MATLAB function solvesbm
A.1.5.

Algorithm 5.9 (Spectral Bundle Method).

Inputs: cost matrix $C \in \mathbb{S}_n$, adjoint operator $A^T$, vector $b \in \mathbb{R}^m$, starting point $y^0 \in \mathbb{R}^m$, stopping parameter $\epsilon \geq 0$, descent parameter $\kappa \in (0, 1)$, weight $u > 0$.

0. (Initialization) Set $k = 0$, $\hat{y}^0 = y^0$, compute $f(y^0)$ and determine $\hat{W}^0$ (oracle).

1. (Find trial point) Compute $W^{k+1}$ and $y^{k+1}$ by solving (Q) (Proposition 5.6).

2. (Stopping criterion) If $f(\hat{y}^k) - f_{W^{k+1}}(y^{k+1}) \leq \epsilon(|f(\hat{y}^k)| + 1)$ then stop.

3. (Evaluation) Compute $f(y^{k+1})$ and find a subgradient $W^{k+1}_S$ (oracle).

4. (Descent test) If $f(\hat{y}^k) - f(y^{k+1}) \geq \kappa [f(\hat{y}^k) - f_{W^{k+1}}(y^{k+1})]$ then set $\hat{y}^{k+1} = y^{k+1}$ (descent step); otherwise set $\hat{y}^{k+1} = \hat{y}^k$ (null step).

5. (Update the model) Choose $\hat{W}^{k+1} \supset \{W^{k+1}, W^{k+1}_S\}$ of the form (5.4.2).

6. Increase $k$ by one and goto 2.

Helmberg [3] proved the convergence of Algorithm 5.9 for $\epsilon = 0$. The proof is
divided into several steps. It is first proven that the stopping criterion identifies
optimal solutions correctly if the algorithm stops after a finite number of iterations.
After an analysis of the asymptotic behavior of null steps, it is then shown that unless
the current stability center is optimal, a descent step will eventually be triggered after
a finite number of iterations. It is finally proven that the sequence of objective values
at the stability centers satisfies $f(\hat{y}^k) \downarrow \inf_{y \in \mathbb{R}^m} f(y)$ in the case of infinitely many
descent steps.

Theorem 5.10 (Convergence). Let $\{\hat{y}^k\}$ be the sequence of stability centers generated
by Algorithm 5.9. Then either $\hat{y}^k \rightarrow \hat{y} \in \text{Argmin}_{y \in \mathbb{R}^m} f(y)$, or $\text{Argmin}_{y \in \mathbb{R}^m} f(y) = \emptyset$ and $\|\hat{y}\| \rightarrow \infty$. In both cases $f(\hat{y}^k) \downarrow \inf_{y \in \mathbb{R}^m} f(y)$.

Proof. See Helmberg [3], Theorem 5.3.6.

The proof of convergence shows that (5.4.11) is the only requirement for a sequence
of null steps to produce a descent step over time. It is also interesting to note that
the solution $W^k$ of the quadratic subproblem can be related to the optimal solutions
of the primal problem (PSDP) (see Helmberg [3], Theorem 5.3.8). In particular, if
Algorithm 5.9 stops after a finite number of steps, say at iteration $k = K$, then
$W^{K+1}$ belongs to the set of optimal solutions of the primal semidefinite program
corresponding to (EOP), i.e.

$$W^{K+1} \in \text{Argmax} \{\langle C, X \rangle : AX = b, \text{tr } X = a, X \succeq 0\}.$$
5.4.4 Improvements

As this paper is only a humble semester project, our MATLAB implementation solvesbm A.1.5 of the Spectral Bundle Method is by far not as effective as it could be. There are indeed many improvements and modifications that could be made to it in order to save time and memory. Here we briefly state the different aspects that could be improved.

Firstly, it may be useful to restart the Lanczos process after $n_L$ steps by setting the new starting vector to the Lanczos vector corresponding to the maximal eigenvalue of the tridiagonal matrix. A small parameter $n_L$ will be of interest when the matrix-vector multiplication is computationally cheap, otherwise it is preferable to choose a larger parameter. This topic is discussed by Helmberg [4], who proposes different heuristics for choosing the parameter $n_L$. Step 4 of Algorithm 5.9 may also be slightly modified in order to save time by stopping eigenvalue computations early when the current estimate is already good enough to prove that the current iterate will result in a null step (see Helmberg [4]).

Secondly, the weight $u$ of the augmented model (5.4.4) is usually updated during the algorithm for efficiency reasons. Since this parameter controls the distance of the new iterate with respect to the center of stability, it has to adapt to the local geometry of the objective function. Indeed, a large value of $u$ penalizes the distance from the stability center and therefore allows only small steps. In this view, a sequence of descent steps would for example indicate that $u$ should be decreased in order to take larger steps. In the same vein, if a long sequence of null steps precedes a descent step, a larger value of $u$ might be better. Several rules for updating parameter $u$ are proposed by Helmberg [4], none of them being known to be the best.

Thirdly, rather than solving the quadratic subproblem (Q) with the MATLAB toolbox SeDuMi, it may be faster to solve it with an appropriate interior point method exploiting the structure of the problem. As aforementioned, such a method is described by Helmberg [3] in Section 5.5. See also Helmberg and Rendl [6]; Helmberg and Kiwiel [5] for a full description of the interior point code.

Finally, the number of Lanczos vectors to add and to keep in the bundle may also be dynamically updated at each iteration. Helmberg [4] proposes a heuristic which chooses four parameters at each iteration for controlling the model update.
6 Resolution of the Max-Cut Problem

Let us assume that we are given the Laplace matrix $L$ of a graph on $n$ vertices. In this section, we explain how to compute an approximation of the optimal solution to the original \textsc{max-cut} problem in algebraic form

\[
\text{(AMC)} \quad \max_{x \in \{-1,1\}^n} x^T L x
\]

from an optimal solution of its eigenvalue relaxation

\[
\text{(E)} \quad \min_{y \in \mathbb{R}^n} \left[ n \lambda_{\text{max}}(L - \text{Diag}(y)) + e^T y \right] = f(y)
\]

Assume that $y_E \in \mathbb{R}^n$ is an optimal solution of (E) obtained by the \textsc{matlab} function \texttt{solvebm} A.1.5, which is an implementation of Algorithm 5.9 (i.e., the Spectral Bundle Method) for the particular case of the \textsc{max-cut} problem.

By construction of the eigenvalue relaxation, we know that $\gamma := f(y_E)$ is an upper bound for the optimal value of (AMC). The goal is now to find a cut vector $x \in \{-1,1\}^n$ whose objective value $x^T L x$ is very close to $\gamma$. This is done in several steps.

6.1 Extraction of a Dual Optimal Solution

To begin with, we use the equivalence between the eigenvalue relaxation (E) and the dual semidefinite relaxation

\[
\text{(D)} \quad \min_{y \in \mathbb{R}^n} e^T y \quad \text{s.t.} \quad \text{Diag}(y) - Z = L, \\
\text{Diag}(y) - Z \succeq 0, \quad y \in \mathbb{R}^n
\]

in order to compute an optimal solution of (D) from $y_E$. This equivalence is ensured by Proposition 5.1, whose proof tells us that

\[
\hat{y} := y_E + \lambda_{\text{max}}(L - \text{Diag}(y_E)) e \in \mathbb{R}^n
\]

is a feasible solution of (D) with objective value $e^T \hat{y} = \gamma$. As a consequence, $(\hat{y}, \hat{Z})$ with $\hat{Z} := \text{Diag}(\hat{y}) - L$ is an optimal solution of (D). This first step is implemented in the \textsc{matlab} function \texttt{shifty} A.2.1.

6.2 Extraction of a Primal Optimal Solution

In the next step, we use the strong duality between the dual semidefinite relaxation (D) and the primal semidefinite relaxation

\[
\text{(P)} \quad \max \quad \langle L, X \rangle \\
\text{s.t.} \quad \text{diag}(X) = e, \\
X \succeq 0
\]

in order to compute an optimal solution of (P) from $(\hat{y}, \hat{Z})$. Strong duality is ensured by Theorem 4.2 and the fact that the dual problem (D) has strictly feasible solutions (cf. proof of Proposition 5.1).
The goal is to find a matrix $\hat{X} \in S_n$ with objective value $\langle L, \hat{X} \rangle = \gamma$ and satisfying the constraints $\text{diag}(\hat{X}) = e$ and $\hat{X} \succ 0$. The condition on the objective value forces the duality gap $\langle \tilde{X}, \tilde{Z} \rangle$ to be zero. Since $\tilde{Z}$ is positive semidefinite and $\hat{X}$ must be positive semidefinite, this requires the columns of $\hat{X}$ to be in the nullspace of $\tilde{Z}$. Let $N$ be a basis for the nullspace of $\tilde{Z}$ and assume it has dimension $k \leq n$, i.e. $N$ is an $n \times k$ matrix. We are now looking for a $k \times n$ matrix $O$ such that $NO$ is symmetric and satisfies $\text{diag}(NO) = e$.

For the sake of convenience, let us denote the $i$-th row of $N$ by $N_i$ and the $i$-th column of $O$ by $O^i$, for $i = 1, \ldots, n$. With respect to this notation, the constraint $\text{diag}(NO) = e$ is equivalent to the system

$$N_iO^i = 1, \quad i = 1, \ldots, n. \quad (6.2.1)$$

Moreover, requiring $NO$ to be symmetric amounts to require

$$N_iO^i - N_jO^j = 0, \quad j = 1, \ldots, n-1, \quad i = j + 1, \ldots, n. \quad (6.2.2)$$

Note that we have $nk$ variables (i.e., the number of entries in $O$) and that $(6.2.1)$ and $(6.2.2)$ respectively yield $n$ and $\frac{(n-1)n}{2}$ equations. Thus, unless $k > \frac{n+1}{2}$, there are more equations than variables and the matrix $O$ can be determined. We express (6.2.1)-(6.2.2) as a big linear system

$$A \text{vec}(O) = b, \quad (6.2.3)$$

where $\text{vec}(\cdot) : M_{k,n} \to \mathbb{R}^{kn}$ is the usual vectorization operator stacking the columns of a matrix on top of each other, $A$ is an $\frac{n(n+1)}{2} \times kn$ matrix and $b \in \mathbb{R}^{\frac{n(n+1)}{2}}$. More precisely, we have

$$A = \begin{bmatrix}
N_1 & 0 & 0 & \cdots & 0 \\
0 & N_2 & 0 & \cdots & 0 \\
\vdots & \ddots & \ddots & \cdots & \vdots \\
0 & 0 & \cdots & N_n \\
N_2 & -N_1 & 0 & \cdots & 0 \\
N_3 & 0 & -N_1 & \cdots & 0 \\
\vdots & \ddots & \ddots & \ddots & \vdots \\
N_n & 0 & \cdots & -N_1 \\
0 & N_3 & -N_2 & \cdots & 0 \\
\vdots & \ddots & \ddots & \ddots & \vdots \\
0 & N_n & 0 & \cdots & -N_2 \\
\vdots & & \cdots & & \vdots \\
0 & 0 & \cdots & \cdots & N_{n-1} \\
0 & 0 & \cdots & \cdots & 0
\end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix}
1 \\
1 \\
\vdots \\
1 \\
0 \\
0 \\
\vdots \\
0 \\
0 \\
\vdots \\
0 \\
\vdots \\
0 \\
\vdots \\
0 \\
0 \\
\vdots \\
0
\end{bmatrix}.$$ 

We can then solve system (6.2.3) easily, for example using a QR-factorization of $A$. Transforming the obtained solution $\text{vec}(O)$ back to a $k \times n$ matrix $O$, we eventually get the desired matrix $\hat{X} := NO$. This second step is implemented in the MATLAB function `dual2primal A.2.2`. 

27
6.3 Best Rank One Approximation

As we derived the primal semidefinite relaxation (P) in Section 4.1, we dropped the rank one constraint \( \text{rank}(X) = 1 \) in the original algebraic formulation of the max-cut problem (4.3.1). As a consequence, the primal optimal solution \( \hat{X} \) that we derived is not necessarily of rank one. In order to deal with this problem and with the purpose of extracting an optimal cut more easily, we compute the best rank one approximation of \( \hat{X} \). Using the fact that \( \hat{X} \) is a symmetric matrix, an SVD decomposition of \( \hat{X} \) (cf. Theorem 2.2) reads \( \hat{X} = U\Sigma U^T \). Let \( u \in \mathbb{R}^n \) be the first column of \( U \). Then the matrix

\[
\hat{X}_1 := \sigma_{11}uu^T
\]

is the best rank one approximation of \( \hat{X} \) by Theorem 2.3. This step is implemented in the MATLAB function `bestrank1approx` A.2.3.

6.4 Extraction of an Optimal Cut

Now that we have the best rank one approximation \( \hat{X}_1 \) of \( \hat{X} \), it remains to determine the corresponding cut. Since \( \hat{X}_1 \) is of rank one, there exists a vector \( z \in \mathbb{R}^n \) such that \( \hat{X}_1 = zz^T \). We can then extract a cut vector \( x \in \{-1, 1\}^n \) from \( z \) by setting

\[
x_i := \begin{cases} 
1 & \text{if } z_i \geq 0 \\
-1 & \text{if } z_i < 0 
\end{cases} \quad i = 1, \ldots, n,
\]

which finally yields the cut \( \delta(S) \) associated to the set of vertices

\[
S = \{ i \in \{1, \ldots, n\} : x_i = 1 \}.
\]

This last step is implemented in the MATLAB function `extractcut` A.2.4.
7 Numerical Experiments

As mentioned in subsection 5.4.4, there are many things that could be done to improve the efficiency of our algorithm implemented in the MATLAB function \texttt{solvesbm}. In particular, the weight \( u \) of the augmented model and the parameters governing the update of the bundle (in our case, only the upper bound \( r_{\text{max}} \) for the bundle size) should be dynamically updated at each iteration. As a consequence, it might seem pointless or unfounded at first sight to perform numerical experiments on the running time of \texttt{solvesbm} with these two parameters being fixed throughout the algorithm. However, although our numerical results are not representative of those that can be obtained with an optimal implementation of the spectral bundle method, we can still draw some interesting observations from them.

We begin with an empirical study of the running time of \texttt{solvesbm}. In our numerical experiments, we choose \( \epsilon = 10^{-5} \) as parameter for the stopping criterion (5.4.15). Following Helmberg [4], we also set \( \kappa = 0.1 \) as parameter for the descent test (5.4.10) and \( y^0 = (0, \ldots, 0) \) as the starting point. Since the best choice of parameters \( u \) and \( r_{\text{max}} \) depends significantly on the structure of a graph (i.e., its dimension and density), we perform the numerical experiments on several graphs of the same size in order to compute an average running time.

For each fixed dimension \( n_i = 10^i, \ i = 1, \ldots, 10 \), we generate a set of 10 random complete graphs on \( n_i \) vertices whose edge weights are uniformly distributed in \([0, 1]\). We then solve each of the 10 corresponding eigenvalue relaxations of the \textsc{max-cut} problem \((E)\) with \texttt{solvesbm} and compute the average CPU-time in seconds.

![Figure 7.1: Average CPU-time [s] with respect to the dimension \( n \) of the graph for different combinations of parameters \( u \) and \( r_{\text{max}} \).](image-url)
Figure 7.1 shows how the average CPU-time increases with respect to the dimension of the graphs for different combinations of parameters $u$ and $r_{\text{max}}$, respectively ranging in $\{1, 4, 8\}$ and $\{10, 20\}$. It appears that the average CPU-time is quite sensitive to those two parameters and that it is unclear whether there is a pair $(u, r_{\text{max}})$ ensuring fast convergence for every dimension of the graph.

For the following numerical experiments we set $u = 4$ and $r_{\text{max}} = 10$. In Figure 7.2 we compare the average CPU-time of solvesbm with the average CPU-time needed by SeDuMi for solving the primal and dual semidefinite relaxations of the MAX-CUT problem (i.e., (P) and (D), respectively) for the same sets of random graphs. The scale on the vertical axis is logarithmic, therefore it seems that in all cases the average running time grows exponentially with respect to the dimension of the graphs. Although SeDuMi applied to problem (P) is faster than solvesbm, the difference between their average CPU-times seems to be of the order of a constant. On the contrary, we can see that solvesbm clearly outperforms SeDuMi when the latter it is applied to problem (D).

![Figure 7.2: Comparison of average CPU-times.](image-url)
The spectral bundle method gives a rather good approximation of an optimal solution fairly quickly, but it has a tailing off effect. In other words, the progress that is done in the last iterations, before the stopping criterion terminates the algorithm, is very small. Figure 7.3 shows the error of solvesbm (with respect to the optimal value computed by SeDuMi) at each iteration for a graph on $n = 50$ vertices. The scale on the vertical axis is logarithmic, which shows that the convergence to the optimal solution is much faster in the first iterations than in the last ones.

![Graph with logarithmic scale showing the error in the objective value at each iteration for a graph on 50 vertices.](image)

Figure 7.3: Error in the objective value at each iteration for a graph on 50 vertices.
We now investigate the influence and sensitivity of parameters $u, \kappa$ and $r_{\text{max}}$. As the choice for parameters $u$ and $r_{\text{max}}$ depends on the structure of a graph, in the following we perform numerical experiments on a single instance of (randomly generated) graph and no longer consider the average running time.

Figure 7.4 shows how the CPU-time varies as a function of parameter $u$. In the top part of the figure, the dimension is fixed to $n = 50$ and the upperbound $r_{\text{max}}$ for the bundle size ranges the set $\{10, 15, 20\}$. In the bottom part of the figure, the bundle parameter is fixed to $r_{\text{max}} = 10$ and the dimension $n$ ranges the set $\{20, 40, 50\}$.

We can see that small values of $u$ should be avoided, as it significantly increases the running time of solvemsb. This comes from the fact that too small values of $u$ allow for too large steps (i.e., the distance from the stability center is weakly penalized in the augmented model), which do not provide satisfactory progress in the objective function and therefore lead to several null steps.

Figure 7.4: CPU-time [s] with respect to parameter $u$. 

32
Figure 7.5 shows how parameter $\kappa$ influences the number of iterations (more precisely, the number of descent steps) and the running time of \texttt{solvesbm}. In the top part of the figure, both the number of iterations and the CPU-time appear as functions of $\kappa$. Parameters $u$ and $r_{\text{max}}$ are fixed to $u = 4$ and $r_{\text{max}} = 10$, while the dimension $n$ runs through the set \{20, 50\}. It emphasizes how the CPU-time is naturally related to the number of iterations.

In the bottom part of the figure, the percentage of descent steps is shown as a function of $\kappa$. Again, parameters $u$ and $r_{\text{max}}$ are fixed to $u = 4$ and $r_{\text{max}} = 10$ and the dimension $n$ ranges the set \{20, 50\}. We can see that large values of $\kappa$ tighten the descent test (5.4.10) and therefore allow for less descent steps.
A Matlab codes

A.1 The Spectral Bundle Method

A.1.1 Lanczos with Reorthogonalization

```matlab
function [H,U] = lanczos_orth(A,ns,x)
% Perform ns steps of the Lanczos process with storing the full
% basis U and with reorthogonalization on Hermitian A with starting
% vector x. On return, H is an (ns+1,ns) upper tridiagonal matrix.

% This code will malfunction if A is not square, if x is of different
% dimension from A, if x = 0, or if ns is greater than the dimension
% of A.

x = x / norm(x);

H = zeros(ns+1,ns+1);
n = size(A,1);
% store all vectors
U = x;
beta = 0;
ns = min(n,ns);
for j = 1:ns
    z = A*U(:,end);
    alpha = U(:,end)'*z;
    u = z - U(:,end)*alpha;
    if j > 1,
        u = u - U(:,end-1)*beta;
    end
    % Reorthogonalization
    alphas = U'*u;
    u = u - U*alphas;
    alpha = alpha + alphas(end);
    beta = norm(u);
    u = u / beta;
    U = [U,u];
    H(j,j) = alpha;
    H(j+1,j) = beta;
    H(j,j+1) = beta;
end
H = H(:,1:end-1);
end
```
A.1.2 The Oracle

```matlab
function [fy,V] = oracle(n,y,L,nL)
%ORACLE: Returns the value fy of the objective function and a
%matrix V containing (at most nL) Lanczos vectors of the matrix
%L-Diag(y).
if n <= 100
    % Compute max. eigenvalue of L-Diag(y) using the Matlab eig ...
    function
    [V,D] = eig(L-diag(y));
    [eigval,I] = sort(diag(D),'descend');
    V = V(:,I);
    V = V(:,1:min(n,nL));
    fy = n * eigval(1) + sum(y);
else
    % Compute max. eigenvalue of L-Diag(y) using Lanczos method
    x = rand(n,1);
    [H,U] = lanczos_orth(L-diag(y),nL,x);
    [V,D] = eig(H(1:end-1,:));
    [eigval,I] = sort(diag(D),'descend');
    V = U(:,1:end-1) * V(:,I);
    fy = n * eigval(1) + sum(y);
end
end
```

A.1.3 The Trial Point Finding

```matlab
function [W_new,y_new,V,alpha] = trial_point(n,y,L,P,Wbar,u,r)
%TRIAL_POINT: solves the semidefinite convex quadratic subproblem using
%SeDuMi.
e = ones(n,1);
% Solve (Q) with SeDuMi
cvx_begin sdp quiet
variable V(r,r) symmetric
minimize( pow_pos(norm(e-diag(P*V*P' + (n - trace(V))*Wbar)),2)/(2*u) - ...
    sum(sum((P*V*P' + (n - trace(V))*Wbar) .* ...
    (L-diag(y)))) - sum(y) )
trace(V) <= n
V >= 0
cvx_end
alpha = n - trace(V);
W_new = P*V*P' + alpha*Wbar;
y_new = y + (diag(W_new)-e)/u;
end
```
A.1.4 The Model Update

```matlab
function [P,Wbar,r] = model_updating(P,Wbar,r,V,alpha,V_new,rmax)

%MODEL_UPDATING: updates the bundle matrix and the aggregate matrix

[Q,D] = eig(V);
[d,I] = sort(diag(D),'descend');
Q = Q(:,I);
D = diag(d);

% aggregation tolerance
ta = 0.01;

% maximum number of Lanczos vectors to add in the bundle
nA = 7;

% minimum number of columns to maintain in the bundle
rmin = min(r,max(0,rmax-nA));

N = rmin;
for i=(rmin+1):r
    if (d(i) >= ta*d(1)) && (i >= rmin)
        N = i;
    end
end

% number of Lanczos vectors to add in the bundle
radd = min(r,rmax-N);

Q1 = Q(:,1:N);
D1 = D(1:N,1:N);
Q2 = Q(:,(N+1):end);
D2 = D((N+1):end,(N+1):end);

% new aggregate matrix
Wbar = (P*Q1*V_new(:,1:radd)+alpha*Wbar)/(trace(D2)+alpha);

% new bundle matrix
P = orth([P*Q1,V_new(:,1:radd)]);

% new bundle size
r = size(P,2);
end
```

A.1.5 The Algorithm

```matlab
function [sbm] = solvesbm(n,L,epsilon,kappa,u,rmax,maxit)

%SOLVESBM: Solves the eigenvalue optimization relaxation of the Max-Cut problem for the graph corresponding to the laplace matrix L using the Spectral Bundle Method

% Step 0 (Initialization)

t = cputime;
nL = 100; % number of steps for Lanczos
y = zeros(n,1); % starting point for the iterates

[fy0,V0] = oracle(n,y,L,nL);

Wbar = eye(n)/n; % initial aggregate matrix
r = min(rmax,size(V0,2)); % initial bundle size
P = V0(:,1:r); % initial bundle matrix
```
fy = [fy0]; % store the objective function values
dsteps = 0; % counter for descent steps
k = 1; % counter for iterations

% Define the linear minorant f_W(y)
f = @(W,y) (sum(sum((L-diag(y)).*W)))+sum(y));

while k \leq \text{maxit}
% Step 1 (Trial point finding)
[W_new,y_new,V,alpha] = trial_point(n,y,L,P,Wbar,u,r);
% Step 2 (Stopping criterion)
if (fy(k)−f(W_new,y_new)) \leq \epsilon (abs(fy(k))+1)
  break;
end
% Step 3 (Evaluation)
[fy_new,V_new] = oracle(n,y_new,L,nL);
% Step 4 (Descent test)
if (fy(k)−fy_new) \geq \kappa (fy(k)−f(W_new,y_new))
  dsteps = dsteps + 1;
y = y_new; % Update the center of stability
fy(k+1) = fy_new; % Store the new objective value
else
  % Null step
  fy(k+1) = fy(k); % Store the new objective value
end
% Step 5 (Model updating)
[P,Wbar,r] = model_updating(P,Wbar,r,V,alpha,V_new,rmax);

% Next step
k = k + 1;
dsteps = dsteps/(k−1);

sbm.t = cputime−t; % cpu time in seconds
sbm.y = y; % optimal solution y
sbm.fy = fy; % objective function values
sbm.optval = fy(end); % optimal value
sbm.dsteps = dsteps; % fraction of descent steps
if (k−1) == \text{maxit}
  sbm.iter = 0; % SBM has reached the max. number of iterations
else
  sbm.iter = (k−1);
end
end
A.2 Resolution of the Max-Cut Problem

A.2.1 Extraction of a Dual Optimal Solution

```matlab
function [y,Z] = shifty(sbm_y,L)

%SHIFTY: Extraction of a dual optimal solution
y = sbm_y + max(eig(L - diag(sbm_y)));
Z = diag(y) - L;
end
```

A.2.2 Extraction of a Primal Optimal Solution

```matlab
function [X,gap] = dual2primal(y,Z)

%DUAL2PRIMAL: Extraction of a primal optimal solution
tol = 1e-6;
n = length(y);
[V,D] = eig(Z);
N = [];
for i=1:n
    if D(i,i) < tol
        D(i,i) = 0;
        N = [N V(:,i)];
    end
end
Z = V*D/V;
k = size(N,2);
% Construction of the system of constraints
b = [ones(n,1);zeros(n*(n-1)/2,1)];
A = sparse(n,n*k);
for j=1:n
    % Enforce constraint X_jj = 1
    A(j,(k*(j-1)+1):j*k) = N(j,:);
    for i=(j+1):n
        % Enforce constraint x_ij = x_ji
        C = sparse(1,[(k*(j-1)+1):j*k (k*(i-1)+1):i*k],[N(i,:) ... -N(j,:)],1,n*k);
        A = [A;C];
    end
end
vecO = A\b;
O = reshape(vecO,k,n);
X = N*O;
gap = sum(sum(Z.*X));
end
```
A.2.3  Best Rank One Approximation

```matlab
function [X1] = bestrank1approx(X)
%BESTRANK1APPROX: Best rank one approximation of X
n = size(X,2);
[U,S,V] = svd(X);
% Keep only the information from the largest singular value
X1 = U(:,1) * S(1,1) * V(:,1)';
end
```

A.2.4  Extraction of an Optimal Cut

```matlab
function [x] = extractcut(X)
%EXTRACTCUT: Extraction of the optimal cut
n = size(X,2);
[U,S,V] = svd(X);
% Extract an incidence vector from the first column of U
for i=1:n
if U(i,1) ≥ 0
    x(i) = 1;
else
    x(i) = -1;
end
end
end
```
References


