Algorithms for Solving Nonconvex Block Constrained Quadratic Problems

Shimrit Shtern* Aharon Ben-Tal†

Abstract

Non-convex Quadratically Constrained Quadratic Programs with block-separable convex constraints are generally NP-hard. These problems appear in many applications such as estimation and control, complex unimodular programming, and MAX-CUT type problems. SDP relaxation is the best known upper bound approximation for this problem. We suggest the Block Optimal Ascent (BOA) algorithm to obtain lower approximation. The algorithm uses block hidden convexity to apply block alternating minimization and we prove it has a sublinear rate of convergence. A better approximation can be obtained using a 'good' initial point, based on the novel approach of using the SDP relaxation solution. A quantitative study shows that using BOA from this initial point has superior performance.

1 Introduction

In this paper we consider the following special case of a *quadratic constrained quadratic program* (QCQP) in which the objective is to maximize a convex quadratic function and the constraints are block-separable:

$$\begin{aligned} \max_{x_i\in\mathbb{R}^{n_i},\ i=1,\ldots,m} & \left\|\sum_{i=1}^{m}\tilde{A}_i x_i\right\|^2 \\ s.t. \quad & \|x_i\|^2 \le 1 \quad \forall i=1,\ldots,m, \end{aligned} \tag{PN}$$

where $x=(x_1,x_2,\ldots,x_m)$ is a partition of the vector $x\in\mathbb{R}^n$, with $x_i\in\mathbb{R}^{n_i}$ such that $\sum_{i=1}^m n_i = n$.

Problem (PN) captures several important applications. Firstly it is a generalization of the MAX-CUT problem. This in itself implies that (PN) is NP-hard. Other applications include the problem

*William Davidson Faculty of Industrial Engineering and Management, Technion—Israel Institute of Technology(shimrits@tx.techniona.ac.il).

†William Davidson Faculty of Industrial Engineering and Management, Technion—Israel Institute of Technology(abental@ie.technion.ac.il).

of finding the *minimal energy spin state* of alloys called 'spin-glass',*complex unimodular programs* used to find a linear encoding which maximize signal to noise ratio, and *robust estimation*. Robust linear estimation, in particular tracking problems, are modeled as the following minimax problem

$$\min_K \max_x F(K,x) = \min_K H(K)$$

where x is the noise vector, K is a filter matrix and F is a measure describing the estimation error. In a linear system, the inner maximization problem of finding the worst case noise vector for a given K is of the structure (PN). This inner problem serves as an oracle that for a fixes K provides information on $H(K)$ and its gradient (subradient). In such applications, numerous calls to this oracle may be needed, and therefore, both accuracy of the solution and computational efficiency are essential. All the above applications are extensively presented in section 3.

Historically, QCQP problems were addressed computationally as early as the 1970's by Branch and bound methods. Branch and bound method dealing with concave objectives and convex feasible set are described in [25, 10]. Al-Khayyal et al. [1] suggested a branch and bound algorithm specifically adapted to non-convex QCQP. Al-Khayyal et al. represent the QCQP as a bi-linear problem with added linear equalities, and use rectangular polygons to obtain a piece-wise linear convex envelope of these bi-linear functions. The authors do not assume a convex feasible set and therefore bound this set as well as the objective function at each stage. However, problem (PN) has a convex feasible set and thus renders such a relaxation redundant. That is, instead of solving a LP at each stage, one may solve a conic quadratic problem, with tighter envelopes. An extensive review of the latest developments in non-linear global optimization is given in [17]

The major disadvantage of branch and bound type algorithms is their high computational complexity. We already stated that problem (PN) is generally NP-hard, and so the running time of any algorithm which converges to the optimal solution is exponential in the problem size. The aforementioned branch and bound algorithms need to solve $O\left(\left(\frac{1}{\epsilon}\right)^n\right)$ (worst case) convex problems (with increasing number of constraints) to obtain a solution with accuracy ϵ. Branch and bound methods apply smart branching and pruning heuristics to reduce the actual running time of the algorithm, but can not guarantee better worst case results. Furthermore, each iteration requires a solution of

an optimization problem, which, while convex, might still incur increasing computational cost. In general, these kind of methods are not practical for large scale problems and so the need arose for a different approach which provide polynomial time approximation for the problem, instead of a global solution.

One such endeavor was initiated by Goemans and Williamson [18], and relied on the SDP relaxation. They proved that in the case of MAX-CUT with positive weights, the SDP relaxation gives a 0.87856 approximation to the problem. Later works ([47, 34, 33]) proved approximations given by the SDP relaxation under different assumptions on the problem's parameters. The most relevant to problem (PN) was Nesterov and Ye's result [35, 46] showing that given $\mathcal{A}$ is PSD and $\mathcal{B}_i$ commute than the SDP relaxation provides a $\frac{2}{\pi}$ approximation and Nesterov later showed [34] that this approximation can be improved to 0.6712 by using additional information. The main issue with the SDP relaxation is that the approximation is usually attained by a randomization scheme, and a polynomial time de-randomization is not always available, so the lower bound guarantee can not be found deterministically . Moreover, in SDP relaxations, we optimize over matrices, instead of vectors, and therefore the dimension of the problem increases quadratically with the size of the original vector x. This fact makes it hard to solve the relaxation, although it is a convex problem. We will show that, due to the specific structure of problem (PN), its SDP relaxation can be solved efficiently with a tailored interior point method. This method, which is an adaptation of an algorithm suggested by Helmberg et al. [23], performs well even when problem (PN) is large, and converges in a linear rate.

Other approaches for obtaining a deterministic solution lie in algorithms designed to find a point which satisfies some optimality conditions, and which utilize the structure and properties of problem (PN). Algorithms such as the *proximal gradient (projected gradient)* method [31, 41] converge to a stationary point, but usually can not provide an approximation. The separable block structure of the constraints, in problem (PN) allows to implement block variations of this algorithm, and may help avoiding some non optimal stationary points. Alternating minimization algorithms, also known as Gauss-Seidel algorithms (see [12]), can also utilize the block structure of the problem. These algorithms are known to converge for convex problems [12, 28, 45], but may not converge in

general [40].

In this paper we present the *Block Optimal Ascent* (BOA) algorithm, a block alternating optimization algorithm specifically designed for problem (PN). The optimization over each block is possible due to a hidden convexity property of the block-wise problems, which allows to convert these non-convex problems to convex ones, and solve them efficiently [44, 5]. We show that the sequence generated by BOA has limits points which satisfy global necessary optimality conditions as well as providing an approximation for problem (PN). Furthermore, we utilize the framework of the proximal gradient theory to show that the BOA algorithm has sub-linear rate of convergence. Lastly, we present a novel approach of finding an initial point for BOA, which is based on the SDP relaxation solution. By using the spectral decomposition of this solution we construct a 'good' initial point, in the sense of guaranteeing an improved approximation of problem (PN). A computational study, presented in Section 9, demonstrates the superior performance of the *Lagrange Guided Ascent* (LGA) algorithm, which combines the SDP based initial point with the BOA algorithm.

Paper organization. Section 2 introduces notations regrading the general structure of QCQP problems and specifically that of problem (PN). Section 3 presents several applications for problem (PN). Section 4 details the cases where problem (PN) is solvable, and the algorithms used to solve these cases. Section 5 elaborates on the use of SDP relaxation for QCQP problems in general and problem PN in particular. Section 6 details the use of the proximal gradient method and its block variants, including proof of convergence. Section 7 presents BOA - the new alternating block optimization algorithm for problem (PN), and its variations (cyclic, greedy and randomized) including proof of the rate of convergence. Section 8 presents a scheme for determining an initial point, based on the SDP relaxation. Finally, in Section 9, we conclude with a computational study which demonstrates our theoretical results.

2 Concave Block-Separable QCQP

QCQP is of the general form:

$$\begin{aligned} &\max_{x\in\mathbb{R}^n} \; x^T\mathcal{A}x \\ &s.t. \quad x^T\mathcal{B}_i x \le \beta_i \quad \forall i=1,\dots,m. \end{aligned} \tag{QCQP}$$

where $\mathcal{A}$, and $\{\mathcal{B}_i\}_{i=1}^m$, are some symmetric matrices. This problem is well known in literature and has various uses [49]. In general QCQP is not convex and computationally it is an NP-hard problem [33], although it is solvable in specific cases. Here, we consider the case where $\mathcal{A}$ is PSD, $\beta_i > 0$ and the constraints are separable, i.e. each constraint can be reformulated as $x^T\mathcal{B}_i x = x_i^T B_i x_i \le \beta_i$ where $\{B_i\}_{i=1}^m$ are PD matrices, and $\{x_i\}_{i=1}^m$, is a partition of vector x. In this case we can use the decomposition $\mathcal{A} = E^T E$ where $E = [A_1,\dots,A_m]$. We also denote I_i the matrix which defines partition element i such that $x_i = I_i x$ and $x = \sum_{i=1}^m I_i^T x_i$. We also denote $\tilde{E} = [\tilde{A}_1,\dots,\tilde{A}_m] = \sum_{i=1}^m \tilde{A}_i I_i$, $\tilde{\mathcal{A}} = \tilde{E}^T\tilde{E}$. Using the isomorphic transformation: $\tilde{x}_i = B_i^{1/2}x_i/\sqrt{\beta_i}$ (recall $\beta_i > 0$), and letting $\tilde{A}_i = \sqrt{\beta_i}A_i B_i^{-1/2}$. Consequently the constraints become norm constraints and the objective function is reformulated as:

$$\max_{\tilde{x}_i\in\mathbb{R}^{n_i}\ \forall i=1,\dots,m} \left\|\sum_{i=1}^m \tilde{A}_i\tilde{x}_i\right\|^2 = \max_{x\in\mathbb{R}^n} \left\|\tilde{E}\tilde{x}\right\|^2 = \max_{x\in\mathbb{R}^n} \tilde{x}^T\tilde{\mathcal{A}}\tilde{x}$$

.

Problem (PN) is a *maximization* of a convex quadratic objective function over separable block-wise norm type constraints, which define a convex, compact set. Therefore there exists an optimal solution where all the constraints are active. As a result each constraint in problem (PN) can be written as either an equality or inequality constraint.

3 Some Applications of Problem (PN)

In this section we review several well known problems which are (or can be) modeled as problem (PN).

3.1 MAX-CUT

The well known MAX-CUT problem is described as finding a partition of graph vertices which maximizes the sum of weights on edges connecting vertices which lie in different partition elements.

The corresponding optimization problem is:

$$\begin{aligned} &\max_{x} \ \frac{1}{4}\sum_{i\neq j} w_{i,l}(1-x_i x_j) \\ &s.t. \quad x \in \{-1,1\}^n \end{aligned} \tag{1}$$

Since $x_i^2 = 1$ one can rewrite the objective as $x^T W x$ where W is the problem's Laplacian, which is a PSD matrix when the weights w are non-negative. The final formulation is then given by:

$$\begin{aligned} &\max_{x} \ x^T W x \\ &s.t. \quad x_i^2 \leq 1, \quad i = 1,\ldots,n. \end{aligned} \tag{2}$$

This is exactly a concave QCQP over a convex set, where the $m = n$ and each of the blocks are of size 1.

MAX-CUT problems have many applications, e.g., via minimization in the layout of electronic circuits, physics of disordered systems, and network reliability, some of which are presented in [3].

3.2 The Spin-Glass Problem

One of the interesting physics applications which generalizes the MAX-CUT problem is the spin-glass problem presented in [3]. Spin glass are alloys with magnetic properties which peak at certain temperatures. The model represents each atom spin as a three dimensional unit length vector $S_i \in \mathbb{R}^3$, two magnetic atoms will create a magnetic interaction J_{ij} which is dependent on atoms distance, and other material properties. The energy between each two atoms is then given by $H_{ij} = S_i^T J_{ij} S_j$ and additional energy is generated by an exterior magnetic field $F \in \mathbb{R}^3$ which induces additional energy $F^T U S_i$ on each atom. At $0°K$ the spin glass attains minimal energy configuration which may be obtained by minimizing the energy of the system given by

$$H(\omega) = -\sum_{i<j} S_i^T J_{i,j} S_j - \sum_i F^T U S_i.$$

The MAX-CUT like integer representation of this problem, assumed that U and J_{ij} are scalars and transforms each vector S_i into scalar $s_i \in \{+1, -1\}$ and the field F into a scalar f. This representation is, in fact, a simplification of the model. This simplification is referred to as the "Ising-spin" and is correct only for some substances. The original model is a concave QCQP with a unique matrix structure since each block in the objective function matrix is in fact a 3×3 diagonal matrix, with off-diagonal equal to J_{ij} for $i \neq j$, U for blocks which correspond to the exterior interaction and diagonal blocks which are the absolute value row sum. Notice that even if $J_{i,j}$ and U are matrices (variant to rotation in spin) taking large enough diagonal elements will only impact the solution by a constant and still maintains the concave QCQP structure.

3.3 Complex Unimodular Programming

Consider a radar system, as described in [15] and [14], which transmits monotone linearly encoded pulses. The observed signal ν is a linear function of the pulses:

$$\nu = \alpha(q \odot p) + w$$

where q is a unimodular vector, containing unimodular elements and p is the temporal steering vector, $\odot$ is an element wise product of matrices, α is the complex echo amplitude (accounting for the transmit amplitude, phase, target reflectivity, and channels propagation effects), and w is a complex zero mean Gaussian vector with covariance matrix M. We denote the conjugate transpose of vector v as v^* and its conjugate as $\overline{v}$. The SNR (signal to noise ratio) is then given by $|\alpha|^2 c^* R c$ where $R = M^{-1} \odot \overline{(pp^*)}$. The problem, is therefore, maximizing a quadratic function over the unimodular set. Similarly, the CRLB (Cramer-Rao lower bound) is given by

$$CRLB = (2|\alpha|^2 c^* R' c)^{-1}$$

where $R' = (c \odot p \odot u)^* M^{-1} (c \odot p \odot u)$, $u = [0, j2\pi T_r, \ldots, j2\pi(n-1)T_r]$, $j = \sqrt{-1}$ and T_r denotes time resolution. Minimizing CRLB is equivalent to maximizing the quadratic PSD form $c^* R' c$ over the unimodular set.

These examples are particular cases of *Unimodular Quadratic Programs*. These kind of problems arise in various signal processing application and are composed of maximizing a quadratic function

over a unimodular vector in the complex n dimensional space, or formally:

$$\max_{s\in\mathbb{C}^n:|s_i|=1} s^*Rs, \tag{3}$$

where R is a complex PD matrix (PD is often assumed although PSD is possible as well).

The problem can be converted to the following real valued quadratic problem:

$$\begin{aligned} \max_{x\in\mathbb{R}^n, y\in\mathbb{R}^n} \quad & z^TQz \\ s.t. \quad & z = [x^T, y^T]^T \\ & x_i^2 + y_i^2 = 1 \, \forall i = 1, \ldots, n \end{aligned} \tag{4}$$

Where $Q = \begin{bmatrix} \Re(R) & \Im(R)^T \\ \Im(R) & \Re(R) \end{bmatrix}$ and if R is complex PSD matrix then Q is a real PSD matrix, and the equalities for index i can be replaced with norm inequalities so to obtain a concave QCQP problem.

In [48] and [8] the authors showed that the SDP relaxation, which we will discuss later on, for this problem where R is Hermitian is a $\frac{\pi}{4}$ approximation. This complex SDP relaxation was used by [29] as well as [19] to approximate the MAX-3-CUT problem.

3.4 Estimation and Control

Consider the following linear system:

$$\begin{aligned} x_t &= F_t x_{t-1} + G_t w_t \\ y_t &= H_t x_t + v_t \end{aligned} \tag{5}$$

Where x_t is the state of the system at time t, w_t is the system noise, y_t is the observation and v_t the observation noise. F_t, G_t, H_t are given parameter matrices which may change with time.

The aim is to construct an estimator, which minimizes a certain measure. This action of estimating a state based on past and present observations is referred to as filtering. The filtering process usually relies on the previous estimator, the known matrices F_t and H_t and a correction term, as follows:

$$\begin{aligned} \hat{x}_t^{t-1} &= F_t \hat{x}_{t-1} \\ \hat{y}_t &= H_t \hat{x}_t^{t-1} \\ z_t &= y_t - \hat{y}_t \\ \hat{x}_t &= \hat{x}_t^{t-1} + K_t(Z_t) \end{aligned} \tag{6}$$

Where $\hat{x}_t^{t-1}$ is the prior estimator at time t, $\hat{x}_t$ is the posterior estimator at time t, $\hat{y}_t$ is the predicted observation. The variable z_t is called the innovation at time t, and is defined by difference between actual and predicted observation. The full history of variable z_τ until time t is denoted by $Z_t = (z_1, z_2, \ldots, z_t)$. The filter $K_t(Z_t)$ is a time dependent function which corrects the prior estimation. The discussion is often restricted to the class of linear estimators $K_t(Z_t) = K_t^T Z_t$. Another important concept is estimation error, we will denote $\delta_t = x_t - \hat{x}_t$ and the size of this deviation $e_t = \|\delta_t\| = \|x_t - \hat{x}_t\|$ gives us some sense of how good the estimation is.

The most common used filter is the Kalman filter which, under certain statistical assumptions over the system and observation noise, gives the minimal mean square error (MSE) linear filter. A different approach is the robust one, which looks at the worst case behavior of the system. This approach tries to find the filter which minimizes this worst case. Assuming that the noise elements of this problem, as well as the initial estimation error, are confined to ellipsoids, we can formulate the problem as follows:

$$\begin{aligned} \min_{K_T} \quad & \max_{\delta_0, \{w_t\}_{t=1}^T, \{v_t\}_{t=1}^T} \|x_T - \hat{x}_T\|^2 \\ s.t. \quad & \delta_0^T \delta_0 \leq \ell^2 \\ & w_t^T Q_t^{-1} w_t \leq \alpha_t^2 \ \forall t = 1, \ldots, T \\ & v_t^T R_t^{-1} v_t \leq \beta_t^2 \ \forall t = 1, \ldots, T \end{aligned} \tag{7}$$

Where matrices Q_t and R_t are positive definite matrices which define the ellipsoids shape and ℓ, α_t and β_t are scalars which define their size. The objective is a quadratic function of the noise variables, and so given a filter K_T finding its worst case realization is equivalent to solving a concave QCQP with block separable constraint. As we showed in Section 2 this problem can be reformulated as problem (PN).

The interesting fact about this problem, which may separate it from other applications is the fact that while the number of constraints m linearly increases with t, the rank of matrix $\mathcal{A}$ remains constant, and is usually small with respect to both m and n, since it is simply the dimension of the state x_t.

4 Solvable Cases of Problem (PN)

4.1 At Most Two Constraints

When problem (PN) has only one constraint it actually transforms to the problem of finding the maximal eigenvalue of matrix $\tilde{\mathcal{A}}$ and its corresponding eigenvector, and has computational cost better than $O(n^3 \log(\frac{1}{\epsilon}))$. This problem is a particular case of the problem known as the *Trust Region Subproblem* (TRSP) described by Sorenson in [44], in which the quadratic objective function is not necessarily homogeneous.

This problem exhibits what is called *hidden convexity*, a term coined by Ben-Tal and Teboulle [9]. It means that although the problem is not convex it can be transformed to a convex problem with the same optimal objective function value. Moreover, the optimal solution of the convex problem can then be translated to an optimal solution of the original problem. The case where both the objective function and the constraint were non-homogeneous quadratic functions, is discussed in [9]. They proved that if both quadratic terms are simultaneously diagonalizable and the linear terms satisfy some further assumptions, then the problem is solvable.

When problem (PN) has two constraints, a solution can be obtained in $O(n^{3.5} \log(\frac{1}{\epsilon}))$ using the SDP relaxation, which we will discuss in the next section. This is a direct result of the existence of an optimal solution to the relaxation with rank r where $r(r+1) \leq 2m$ (See [4, 43, 38]), and so if $m = 2$ then $r = 1$. Also notice that the TRSP can be viewed as a specific case of this problem with $m = 2$, since it can be homogenized by adding a scalar variable s with additional constraint, of $s^2 = 1$.

4.2 Objective Function Matrix of Rank 2

When the row rank of $\tilde{E}$ is small the solution may be achieved by numerically solving for each of the possible row signs. We will demonstrate this for $r = 2$. Denoting $\tilde{E}_1$, $\tilde{E}_2$ as two orthogonal vectors which span the rows $\tilde{E}$. The objective function can be written as $\|[\lambda_1 \tilde{E}_1 x; \lambda_2 \tilde{E}_2 x]\|$ for some

appropriate λ_1 and λ_2, allows reformulating the problem as follows:

$$\begin{aligned} \max_{x,y,z} \quad & y^2 + z^2 \\ s.t. \quad & x^T \mathcal{I}_i x \leq 1 \quad \forall i = 1, \ldots, m \\ & \lambda_1 \tilde{E}_1 x = y \\ & \lambda_2 \tilde{E}_2 x = z \end{aligned} \tag{8}$$

The bounds on y are then defined by $y \in [-y_{max}, y_{max}]$ where $y_{max} = \lambda_1 \sum_{i=1}^{m} \|\mathcal{I}_i (\tilde{E}_1)^T\|$, and similarly for z. Due to symmetry only the case of $y \geq 0$ is considered. Setting $\alpha \leq y \leq \beta$ bounds on the objective function value can be found by solving $\max_{x,z} z^2$ for this domain. This problem is solvable since the quadratic objective $\max_{x,z} z^2$ can be converted into two linear ones: $z_{max} = \max_{x,z} z$ and $z_{min} = \min_{x,z} z$, and so transforming it into a convex problem. The problem than becomes (for the z_{max} case):

$$\begin{aligned} \max_{x,z} \quad & z \\ s.t. \quad & x^T \mathcal{I}_i x \leq 1 \qquad \forall i = 1, \ldots, m \\ & \alpha \leq \lambda_1 \tilde{E}_1 x \leq \beta \\ & \lambda_2 \tilde{E}_2 x = z \end{aligned} \tag{9}$$

The following algorithm solves the problem:

Algorithm 1 Rank 2 algorithm

1: Initialize $y_0 = 0$, $\delta_0 = \sqrt{\epsilon}$, $y_k = \sum_{j=0}^{k-1} \delta_j$ and $\delta_k = -y_k + \sqrt{y_k^2 + \epsilon} \approx \frac{\epsilon}{y_k}$ $k = 0, \ldots, N-1$, where N satisfies $y_N \geq y_{max}$.
2: **for** $k = 1 \rightarrow N$ **do**
3: Solve 9 with $\alpha = y_{k-1}$ and $\beta = y_k$ obtaining solution x_k, $z_k = max(-z_{min}, z_{max})$.
4: Calculate $y_k^* = \lambda^1 \tilde{E}^1 x_k$.
5: Update interval lower bound to $LB_k = y_k^{*2} + z_k^2$
6: **end for**
7: Find $k^* \in \arg\max_k LB_k$
8: Return solution x_{k^*} with value LB_{k^*}.

Proposition 4.1. *Algorithm 1 obtains an ϵ-approximate solution of problem (8), by checking at most $O(\frac{1}{\epsilon})$ points.*

Proof. We begin by showing that the solution obtained is an ϵ-apprixmate solution. Let (x^*, y^*, z^*) be the optimal solution. Assume y^* is in interval $[y_k, y_{k+1}]$ for some $1 \leq k \leq N$. By definition of z_k, $|z^*| \leq |z_k|$ holds. Therefore,

$$y^{*2} + z^{*2} \leq y_{k+1}^2 + z_k^2 = (y_k + \delta_k)^2 + z_k^2.$$

Notice that by construction of y_k and δ_k we have that δ_k is the solution of the quadratic equation $\delta_k^2 + 2y_k\delta_k - \epsilon = 0$ and so

$$(y_k + \delta_k)^2 + z_k^2 - \epsilon = y_k^2 + z_k^2 \leq LB_k \leq LB_{k^*}$$

To compute the number of points needed for this approximation we must first observe that $0 \leq y_k < y_{k+1}$ results in $\delta_{k+1} < \delta_k$. When y_k is very large (and so δ_k very small) we can negate the quadratic term and approximate $\delta_k \approx \frac{\epsilon}{2y_k}$. Therefore, an approximation of the number of points needed can be obtained by $N \leq \frac{y_{max}}{\delta_N}$ which means, for large enough y_{max} that $N \leq \frac{y_{max}}{-y_{max}+\sqrt{y_{max}^2+\epsilon}} \approx \frac{2y_{max}^2}{\epsilon}$. □

This method of course becomes quite cumbersome as the rank of $\tilde{E}$ increases, and so has limited use. However, we will later show that other approximations can be obtained when dealing with low rank $\tilde{E}$.

4.3 Specific Objective Function Structure

Some solvable cases may arise when the matrix $\tilde{\mathcal{A}}$ has a specific structure. One such case is when all the matrices are orthogonally diagonalizable, which happens if and only if the matrices commute [24]. In this case the following conditions holds true:

$$\tilde{\mathcal{A}}\mathcal{I}_i = \mathcal{I}_i\tilde{\mathcal{A}} \quad \forall i = 1, \ldots, m \iff \tilde{A}_j^T\tilde{A}_i = 0 \quad \forall i \neq j \tag{10}$$

In other words, if $\tilde{\mathcal{A}}$ is block wise diagonal the problem becomes block separable. We can solve the problem for each block, using the single constraint solution discussed above. Provided $\tilde{A}_i \neq 0 \ \forall i$, this may only occur if $m \leq r$ (recall r is the rank of matrix $\tilde{\mathcal{A}}$).

Another known solvable case is presented in [47]. Zhang defines objective function matrix $\tilde{\mathcal{A}}$ to be OD-nonnegative, if there exist a sign vector $\sigma \in \{\pm 1\}^n$ such that $\tilde{\mathcal{A}}_{ij}\sigma_i\sigma_j \geq 0 \quad \forall i \neq j$. In this case the SDP relaxation (discussed in the next subsection) has the same objective function value as the original problem, and we can obtain a solution to the original problem in polynomial time using the SDP solution. Verifying whether a matrix is OD-nonnegative takes polynomial time, but in general PSD matrix are not necessarily OD-nonnegative. For example $\begin{bmatrix} 3t & -t & -t \\ -t & 3t & -t \\ -t & -t & 3t \end{bmatrix}$ is clearly a PSD matrix for any $t > 0$ (by diagonal dominance) but since we need three condition for OD-nonnegativity: $\exists \sigma \in \{-1, 1\}^3$ such that $\sigma_1\sigma_2 < 0$, $\sigma_2\sigma_3 < 0$ and $\sigma_1\sigma_3 < 0$, and so $\sigma_1^2\sigma_2^2\sigma_3^2 < 0$ resulting in a contradiction.

5 Approximation for Intractable Cases: SDP Relaxation

In most cases we can not obtain an exact (global) solution to problem (PN). In such cases we can only aim at obtaining an approximate solution.

The most common way of obtaining an upper bound for these kind of problems is by using SDP (Semi-Definite Programming). The well known SDP relaxation to problem (QCQP) (with $\beta_i = 1$) is given by:

$$\begin{aligned} &\max_{\mathcal{X} \in S_+^n} \; Tr(\mathcal{A}\mathcal{X}) \\ &s.t. \quad Tr(\mathcal{B}_i\mathcal{X}) \leq 1 \qquad \forall i = 1, \ldots, m. \end{aligned} \tag{SDP}$$

The approximation obtained by using SDP relaxation for (QCQP) problems has been extensively investigated in the literature. A review of the known results is given in [33] and [27]. The most relevant results pertaining to problem (PN) are Goeman and Williamson 0.87856 approximation for the MAX-CUT problem [18], Nesterov's and Ye's $\frac{2}{\pi}$ approximation for the case matrices $\{\mathcal{B}_i\}_{i=1}^m$ commute and $\mathcal{A}$ is a PSD matrix [35, 46] and Nemirovski et al. $2\ln(2m^2)$ approximation for the case were $\{\mathcal{B}_i\}_{i=1}^m$ are PSD matrices such that $\sum_{i=1}^m \mathcal{B}_i \succ 0$ [33]. These results are based on a randomization scheme, although Nemirovski et al. also present a de-randomization scheme which allows a polynomial time deterministic solution to be obtained.

Furthermore, Shapiro [43] and later Barvinok [4] and Patakai [38], proved that for an SDP with equality constraints there exists a solution with rank r such that $r(r+1) \leq 2m$. This solution can be found in polynomial time using the well known procedure to find a basic feasible solutions in LP (See [4]). As we stated before, this means that the SDP relaxation is tight for $m \leq 2$. We therefore assume, without loss of generality, that the rank of the optimal solution $\mathcal{X}$ to the semidefinite relaxation is always bounded above by $\sqrt{2m}$. This rank upper bound can also be used to improve the rank result attained by Nemirovski et al. [33]. Following the proof, given an optimal solution $\mathcal{X}^*$ with rank r, and a decomposition $\mathcal{X}^* = VV^T$ where V is an $n \times r$ matrix, the matrix $V^T\mathcal{B}_i V$ has rank bounded by r. Consequently, a tighter approximation can be given by $2\ln(2mr) \leq 3\ln(2m)$.

We will now restrict out discussion to problem (QCQP) which satisfies the following assumption.

Assumption 5.1. Matrices $\mathcal{B}_i \succeq 0 \ \forall i$, are of arbitrary rank and $\sum_{i=1}^m \mathcal{B}_i \succ 0$.

Assumption 5.1 implies that since $x^T(\sum_{i=1}^m \mathcal{B}_i)x \leq m$ holds true for any feasible solution and $\mathcal{B}_i \succeq 0 \ \forall i$, the feasible set is both convex and compact. We will call problem (QCQP) for which this assumption holds a *Quadratically Convex Constrained Quadratic Problem* (QCCQP).

Consider the following relaxation of the QCCQP problem.

$$Opt(BR) = \max_{x \in \mathbb{R}^n} \left\{ x^T\mathcal{A}x : x^T\mathcal{B}x = \sum_{i=1}^m x^T\mathcal{B}_i x \leq m \right\} \tag{BR}$$

Since $\mathcal{B} \succ 0$ it is clear that $\mathcal{A}$ and $\mathcal{B}$ are simultaneously diagonalizable (their null space intersection is $\{0\}$, see [20]), and so we can transform the problem to maximizing a quadratic form under a single norm type constraint. As we've seen in section 4 this problem is solvable. Given an optimal solution $\hat{x}$ it is obvious that $\hat{x}/m$ is feasible for the original problem, thus giving a lower bound on the solution, yielding the approximation:

$$Opt(BR) \geq Opt(QCCQP) \geq \frac{1}{m} Opt(BR). \tag{11}$$

However, in the QCCQP case, the approximation obtained by the (SDP) formulation is tighter than that obtained by the (BR), due to the following:

Proposition 5.1. *Given a QCCQP, the SDP relaxation admits the following approximation:*

$$\frac{1}{\rho}Opt(SDP) \leq Opt(QCCQP) \leq Opt(SDP),$$

where $\rho = \min\{r, 2\ln(2mr)\}$ *where* $r = rank(\mathcal{X})$ *such that* $r(r+1) \leq 2m$*, and a feasible solution to QCCQP which achieves this bound can be found in polynomial time.*

Proof. We know that SDP relaxation provides an upper bound for QCCQP. Furthermore we can assume that all constraints are equality constraints, since for each constraint i in the original formulation, adding slack variable s_i^2, does not change the problem structure. Let $\mathcal{X}$ be an optimal solution of the SDP relaxation, with decomposition $\mathcal{X} = \sum_{i=1}^{r} g_i g_i^T$ where $\{g_i\}_{i=1}^r$ are orthogonal, and r is the matrix rank. Let $i^* \in \arg\max_i g_i^T \mathcal{A} g_i$, then g_{i^*} is feasible (since $\mathcal{B}_i \succeq 0$) and as such induces a lower bound of QCCQP with objective function value of at least $1/r$ of the SDP solution. Moreover, according to [33], there is an efficient procedure which finds a feasible solution, with objective value at most $2\ln(2mr)$ times smaller than the SDP value. Finally, by [4], we know that there exists $\mathcal{X}$ with rank that $r(r+1) \leq 2m$ which can be computed efficiently (in polynomial time). Taking the best of these two approximation we achieved the desired result. □

The main conclusion from Proposition 5.1 is that the tightness of the approximation is highly dependent on the rank of $\mathcal{X}$. We will now see how to utilize additional information on problem structure in order to determine the rank of $\mathcal{X}$ and therefore the tightness of the relaxation.

Theorem 5.2. *Let* $\mathcal{A}$*, the objective function matrix of problem QCCQP, be of rank* r*. Then the SDP has a solution* $\mathcal{X}$ *with rank* r *which can be found in polynomial time.*

Proof. The proof is given in appendix A □

There are cases where the structure of a certain SDP relaxation solution $\mathcal{X}$ enables to efficiently construct an optimal solution x to the original problem, although theoretically its approximation is not tight. We will now specify such conditions for problem (PN). In the following proof, for a given a solution $\mathcal{X}$ we denote its the spectral decomposition by $\mathcal{X} = \sum_{j=1}^{n} \lambda_j U^j U^{j^T}$. We further denote by $U_i^j = I_i U^j$ the ith block of the j eigenvector of $\mathcal{X}$.

Proposition 5.3. *Let $\mathcal{X}$ be a solution to the SDP relaxation of (PN) with eigenvectors U^j corresponding to the strictly positive eigenvalues $\lambda_j > 0$ being partition-wise orthogonal, meaning that*

$$U_i^{j^T} U_i^k = 0 \quad \forall i = 1, \dots, m, \forall j, k : \lambda_j \lambda_k > 0$$

then an optimal solution x^* for (PN) can be obtained such that $x^* = \sum_{j=1}^{n} \mu_j U^j$ where μ_j is given by $\mu_1 = \sqrt{\lambda_1}$ and the following recursive formula

$$\mu_j = \begin{cases} \sqrt{\lambda_j} & U^j \tilde{\mathcal{A}} \left(\sum_{k=1}^{j-1} \mu_k U^k \right) \geq 0 \\ -\sqrt{\lambda_j} & otherwise \end{cases} \quad \forall j = 2, \dots, n$$

Proof. From the fact that $U_i^{j^T} U_i^k = 0, \quad i = 1, \dots, m, \ \forall j, k : \lambda_j \lambda_k > 0$ and $\mu_j^2 = \lambda_j$ we have that

$$\begin{aligned} Tr(\mathcal{I}_i \mathcal{X}) &= \sum_{j=1}^{n} \lambda_j Tr(U^{j^T} \mathcal{I}_i U^j) = \sum_{j=1}^{n} \lambda_j \|U_i^j\|^2 \\ &= \sum_{j=1}^{n} \sum_{k=1}^{n} \mu_j \mu_k U_i^{j^T} U_i^k = x^{*T} \mathcal{I}_i x^*, \quad i = 1, \dots, m. \end{aligned} \tag{12}$$

Thus, feasibility of $\mathcal{X}$ implies feasibility of x^*. Moreover, by the construction of μ we can bound the objective function value at point x^* as follows:

$$\begin{aligned} x^{*T} \tilde{\mathcal{A}} x^* &= \sum_{j=1}^{n} \sum_{k=1}^{n} \mu_j \mu_k U^{j^T} \tilde{\mathcal{A}} U^k = \sum_{j=1}^{n} \lambda_j U^{j^T} \tilde{\mathcal{A}} U^j + 2 \sum_{j=1}^{n} \sum_{k=1}^{j-1} \mu_j \mu_k U^{j^T} \tilde{\mathcal{A}} U^k \\ &\geq \sum_{j=1}^{n} \lambda_j U^{j^T} \tilde{\mathcal{A}} U^j = Tr(\tilde{\mathcal{A}} \mathcal{X}) \end{aligned} \tag{13}$$

However, the SDP relaxation optimal value is an upper bound on the objective function value of (PN), for any feasible solution, and specifically x^*. Therefore, by the optimality of $\mathcal{X}$ we have that $Opt(SDP) = Tr(\tilde{\mathcal{A}} \mathcal{X}) = x^{*T} \tilde{\mathcal{A}} x^*$ and we proved that x^* is in fact an optimal solution for (PN) and that $Opt(PN) = Opt(SDP)$. □

Notice that this result is a generalization of the case where $rank(\mathcal{X}) = 1$ for which the construction of x^* is trivially given by the $\lambda_1 U^1$, where λ_1 is the only positive eigenvalue of $\mathcal{X}$.

Solving an SDP, is often practically problematic for large dimensions ($n > 1000$) unless the problem exhibits a certain structure. Helmberg et al. [23] suggested a simple approach for solving SDP problems via a Primal-Dual interior point algorithm. They give an example of an efficient application for the MAX-CUT problem, which has computational complexity of $O(n^3)$ and rate of convergence of $O(\sqrt{n}\log\left(\frac{1}{\epsilon}\right))$. Since our problem is closely related to the MAX-CUT problem we are able extended this method to the SDP relaxation of (PN), maintaining the same computational cost and rate of convergence. For this purpose, in appendix B, we briefly follow the scheme of paper [23], specifying the particular implementation for our application, and detailing some of the proofs corresponding to the case of problem (PN). We will refer to this algorithm as *HRVW-alg.* The actual performance of this algorithm for numerical data will be reported in section 9.

To conclude the discussion on SDP, the most applicable approximation for problem (PN) is Nesterov's $2/\pi$ approximation [34]. We observed that, for general QCCQP, the bound will depend on the rank of the objective function matrix and the number of constraints. Recall that the feasible solution which yields this $2/\pi$ approximation is created by a randomized mechanism, and only the expected objective value of this random solution yields the desired approximation. Since problem (PN) has such a unique structure, we may want to utilize it to deterministically obtain a good feasible solution, which can also serve as a lower bound and an aid to determine the practical tightness of the SDP. We start by presenting the proximal/projected gradient scheme and block proximal gradient for this problem. We then continue by showing how hidden convexity and the SDP relaxation can help us obtain a better solution.

6 The Proximal Gradient Method

The classical *proximal gradient method* (PGM) [31, 41], also known as the *proximal forward-backward* method, is used to minimize non-smooth decomposition models of the form

$$h(x) = f(x) + g(x), \tag{14}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ has a Lipschitz continuous gradient and $g : \mathbb{R}^n \to \mathbb{R} \bigcup \infty$ is non smooth. The PGM generates a sequence $\{x^k\}_{k\in\mathbb{N}}$ by the following:

$$x^{k+1} = prox_{tg}(x^k + t\nabla f(x^k)), \quad t > 0 \tag{15}$$

where the proximal mapping of a function H is defined by

$$prox_H(x) = \arg\min_u \left\{ H(u) + \frac{1}{2}\|u - x\|^2 \right\}. \tag{16}$$

Model (14) is a generalization of any constrained optimization problem:

$$\min_{x\in C} f(x) = \min_{x\in\mathbb{R}^n} \{f(x) + I_C(x)\}.$$

where $g(x) = I_C(x)$ is the indicator function of the feasible set C. In this case the proximal mapping reduces to the orthogonal projection onto set C and the PGM is equivalent to the gradient projection algorithm [26].

Problem (PN) is a specific case of constrained optimization with a non-convex objective and convex separable feasible set. We denote

$$f(x) = -\frac{1}{2}x^T\tilde{\mathcal{A}}x = -\frac{1}{2}\left\|\sum_{i=1}^{m}\tilde{A}_i x_i\right\|^2, \tag{17}$$

where ∇f is Lipschitz continuous with constant $L_f = \left\|\tilde{\mathcal{A}}\right\| = \lambda_{max}(\tilde{\mathcal{A}})$. Here g is the indicator function over the Cartesian product of unit balls, and the projection onto the set is given by the Cartesian products of the projection onto the sets defined by each constraint.

Traditionally, for non-convex problems only subsequence convergence was shown [36]. It is well known that any such limit point is a stationary point of h. The recent work of Bolte et al. [13] presented a general recipe for proving convergence of the entire sequence (*"global convergence"*), for algorithms which satisfy certain conditions. One of the recipe's main ingredients is the assumption that h is semialgebraic, which is easily satisfied in our case. A more precise description of the recipe and the required assumptions can be found in [13]. According to [2] and [13, Proposition 3], the PGM with step size $t < 1/L_f$ converges globally to a stationary point. We will show that

the additional information on the structure of h, which apply to problem (PN), yields that this convergence is true for any value $t > 0$. In the sequel, for convenience, we often use the following notation:

$$G_t(x) = \frac{1}{t}\left(prox_{tg}\left(x - t\nabla f(x)\right) - x\right). \tag{18}$$

Proposition 6.1. *Let f be a concave continuously differentiable function where ∇f is Lipschitz with constant L_f, and let g be a proper lower semicontinuous and convex function. Further assume that function $h = f + g$ is semialgebraic, with minimal value $h^* > -\infty$. Let $\{x^k\}_{k\in\mathbb{N}}$ be the sequence generated by PGM for some constant step size $t > 0$. Then:*

(i) For every $k \in \mathbb{N}$ it holds that

$$\frac{1}{t}\left\|x^{k+1} - x^k\right\|^2 \leq h(x^k) - h(x^{k+1}). \tag{19}$$

(ii) The sequence $\{x^k\}_{k\in\mathbb{N}}$ globally converges to a stationary point and the rate of convergence is given by:

$$\min_{k=0,\ldots,N-1}\left\|G_t(x^k)\right\| \leq \sqrt{\frac{h(x^0) - h^*}{tN}}. \tag{20}$$

Proof. (i) By the definition of the proximal mapping (see (16)) we have that

$$x^{k+1} = prox_{tg}(x^k - t\nabla f(x^k)) = \arg\min_u \left\{tg(u) + \frac{1}{2}\left\|u - x^k + t\nabla f(x^k)\right\|^2\right\}. \tag{21}$$

By optimality conditions at x^{k+1}:

$$0 \in t\partial g(x^{k+1}) + x^{k+1} - (x^k - t\nabla f(x^k)) \tag{22}$$

which, using the notation G_t, can also be written as

$$-(\nabla f(x^k) + G_t(x^k)) \in \partial g(x^k + tG_t(x^k)). \tag{23}$$

Therefore, by the subgradient inequality, for any $v \in \partial g(x^{k+1})$ and specifically for $v = -(\nabla f(x^k) + G_t(x^k))$ we have:

$$\begin{aligned} g(x^{k+1}) &= g(x^k + tG_t(x^k)) \leq g(x^k) + tv^T G_t(x^k) \\ &= g(x^k) - t(\nabla f(x^k) + G_t(x^k))^T G_t(x^k). \end{aligned} \tag{24}$$

Since we are dealing with a concave function f we can use the gradient inequality

$$f(x^k + tG_t(x^k)) \leq f(x^k) + t\nabla f(x^k)^T G_t(x^k). \tag{25}$$

Combining (24) and (25) we have

$$\begin{aligned} h(x^{k+1}) &= h(x^k + tG_t(x^k)) = f(x^k + tG_t(x)) + g(x^k + tG_t(x^k)) \\ &\leq f(x^k) + t\nabla f(x^k)^T G_t(x^k) + g(x^k) - t(\nabla f(x^k) + G_t(x^k))^T G_t(x^k) \\ &= h(x^k) - t\left\|G_t(x^k)\right\|^2. \end{aligned} \tag{26}$$

Using the definition of G_t we obtain (19).

(ii) To prove global convergence we follow the recipe presented in [13]. We have already proven the first step (sufficient decrease). Using (23) and the definition of $v = -\left(\nabla f(x^k) + G_t(x^k)\right)$ yields:

$$\nabla f(x^{k+1}) + v \in \nabla f(x^{k+1}) + \partial g(x^{k+1}) = \partial h(x^{k+1}) \tag{27}$$

and so by the Lipschitz property of ∇f we obtain

$$\begin{aligned} &\left\|\nabla f(x^{k+1}) + v\right| \leq \left\|\nabla f(x^{k+1}) - \nabla f(x^k)\right\| + \left\|G_t(x^k)\right\| \leq \\ &L_f\left\|x^{k+1} - x^k\right\| + \frac{1}{t}\left\|x^{k+1} - x^k\right\| = \left(L_f + \frac{1}{t}\right)\left\|x^{k+1} - x^k\right\| \end{aligned} \tag{28}$$

which achieves the requirements of the second step (subgradient lower bound). The final step is valid simply by the fact that h is semialgebraic, and the global convergence is proven.

The convergence rate is simply given by summing (19) over $k = 0, \ldots, N-1$ and dividing by the number of iterations

$$N \min_{k=0,\ldots,N-1}\left\|x^{k+1} - x^k\right\|^2 \leq \sum_{k=0}^{N-1}\left\|x^{k+1} - x^k\right\|^2 \leq h(x^0) - h(x^N) \leq h(x^0) - h^*.$$

□

Notice that, in our case, $h(x) = f(x)$ for any $x \in C$. Taking, for example, $t = 1/L$ where $L \geq L_f$ we get that

$$\min_{k=0,\ldots,N-1}\left\|x^{k+1} - x^k\right\| \leq \sqrt{\frac{h^* - h(x^0)}{LN}} \leq \sqrt{\frac{mL}{LN}} = \sqrt{\frac{m}{N}}. \tag{29}$$

So we will need to run $O(\frac{m}{\epsilon^2})$ iterations to get $\min_{k=0,\ldots,N-1} \left\|x^{k+1}-x^k\right\| \leq \epsilon$.

Since, in problem (PN), the constraints are separable in the different blocks, we can use the same theory to implement *Block Proximal Gradient Method* (BPGM). First, we define $x_{-i} = [x_1;\ldots;x_{i-1};x_{i+1};\ldots;x_m] \in \mathbb{R}^{n-n_i}$ and similarly $\tilde{A}_{-i} = [\tilde{A}_1,\ldots,\tilde{A}_{i-1},\tilde{A}_{i+1},\ldots,\tilde{A}_m]$ and $C_{-i} = \{x_{-i} : x_j \in C_j,\ \forall j$ Given $x_{-i} \in C_{-i}$ we define $f_i(\cdot|x_{-i}) : \mathbb{R}^{n_i} \to \mathbb{R}$ by

$$f_i(y|x_{-i}) = f(x_1,\ldots,x_{i-1},y,x_{i+1},\ldots,x_m),$$

and similarly $h_i(x_i|x_{-i}) = f_i(x_i|x_{-i})+g_i(x_i)$. Specifically, $f_i(x_i|x_{-i}) = -\frac{1}{2}(\left\|\tilde{A}_i x_i\right\|^2 + 2x_i^T\tilde{A}_i^T\tilde{A}_{-i}x_{-i} + \left\|\tilde{A}_{-i}x_{-i}\right\|^2)$ and it is clear that $\nabla f_i(x_i|x_{-i})$ is Lipschiz continuous with constant $L_{f_i} = \lambda_{max}(\tilde{A}_i^T\tilde{A}_i)$.

The proximal gradient method step can be performed on each block i keeping x_{-i} constant. When performed in a cyclic order it is often referred to as *Alternating Proximal Gradient Method* or *Cyclic Block Proximal Gradient Method* (CBPGM). Rate of convergence of for unconstrained convex optimization problems is discussed in [6]. A constrained version for convex functions using proximal gradient was suggested by Bertsekas and Tseng [11]. Later Bolte et al. [13] showed that for non-convex problems CBPGM converges globally to a stationary point. We will later discuss a non-cyclic scheme and derive a rate of convergence.

In the cyclic version we divide each iteration to m sub-iterations, in each we perform a proximal gradient step on h_i. Let us denote by $z^{k,i} \in \mathbb{R}^n$ the vector after sub-iteration i such that $z^{k,0} = x^k$ and $x^{k+1} = z^{k,m}$. Since we change only one block of the partition at each step we will have that $z_{-i}^{k,i} = z_{-i}^{k,i-1}$ and $z_i^{k,i} = \text{prox}_{t_{k,i}g}(z_i^{k,i-1} + t_{k,i}\nabla f_i(z_i^{k,i-1}|z_{-i}^{k,i-1}))$, where $t_{k,i}$ is the step size. In [13] this method is referred to as PALM and global convergence is proved, provided that $t_{k,i} > L_{f_i}$ (see [13, Theorem 1]).

When f is concave and g is convex we can show, similarly to Proposition 6.1, that global convergence holds true for any constant $t_{k,i}$. More precisely, in this case, we get the following rate of convergence result:

$$\min_{k=0,\ldots,N-1}\max_{i=1,\ldots,m}\left\|G_t^i(z^{k,i-1})\right\| \leq \sqrt{\frac{h(x^0)-h^*}{tN}} \tag{30}$$

where

$$G_t^i(x) = \frac{1}{t}(\text{prox}_{tg_i}(x_i - t\nabla f_i(x_i|x_{-i})) - x_i)$$

and $t_{k,i} \equiv t$ for each k and i. The total step update is then given by the Cartesian product of $\left\{G_t^i(z^{k,i-1})\right\}_{i=1}^n$.

Choosing, for example $t = 1/L$ where $L \geq L_f$ we have that

$$\min_{k=0,\ldots,N-1} \left\|x^{k+1} - x^k\right\| \leq \sqrt{\frac{m}{N}}. \tag{31}$$

It seems that the bound on $\left\|G_t^i(z^{k,i-1})\right\|$ is independent of m. This is not really the case since we have to run m sub-iterations for each iteration meaning that we will need to run at most $O(\frac{m}{\epsilon^2})$ sub-iterations to insure $\min_{k=0,\ldots,N-1} \left\|x^{k+1} - x^k\right\| \leq \epsilon$.

Finally, we will consider a variation of the block algorithm which can utilize parallel computations to decrease the running time of each iteration. Instead of going cyclically between the partition elements at each point we choose one which is furthest from its gradient projection, we call this method *Greedy Block Proximal Gradient Method* (GBPGM). Another version may be to choose i_{k+1}^* such that $f(x^{k+1})$ will be maximal, and the following analysis will remain the same. The update step is defined by:

$$\begin{aligned} i_{k+1}^* &= \arg\max_{i=1,\ldots,m} \left\|\text{prox}_{tg_i}(x_i^k - t\nabla f_i(x_i^k|x_{-i}^k)) - x_i^k\right\| = \arg\max_{i=1,\ldots,m} \left\|G_t^i(x^k)\right\| \\ x_i^{k+1} &= \begin{cases} \text{prox}_{tg_i}(x_i^k - t\nabla f_i(x_i^k|x_{-i}^k)) & i = i_{k+1}^* \\ x_i^k & i \neq i_{k+1}^* \end{cases}, \quad t > 0. \end{aligned} \tag{32}$$

Theorem 6.2. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a continuously differentiable function where ∇f is Lipschitz with constant L_f. Let g_i, $i = 1, \ldots, m$, be proper, lower semicontinuous and convex. Further assume that $h(x) = f(x) + \sum_{i=1}^m g_i(x_i)$ has a minimal value $h^* > -\infty$. Let $\{x^k\}_{k\in\mathbb{N}}$ be the sequence generated by the GBPGM for some constant step size $t > 0$ which is assumed to be bounded. Then:*

(i) For each $k \in \mathbb{N}$ it holds that $\frac{1}{t}\|x^{k+1} - x^k\|^2 \leq h(x^k) - h(x^{k+1})$.

(ii) All limit points of the sequence $\{x^k\}_{k\in\mathbb{N}}$ are stationary points with the same objective function value.

(iii) The rate of sub sequence convergence is given by:

$$\min_{k=0,\ldots,N-1} \left\| G_t(x^k) \right\| \leq \sqrt{\frac{m(h(x^0) - h^*)}{tN}}. \tag{33}$$

Proof. (i) Similarly to Proposition 6.1 (i):

$$h(x^{k+1}) \leq h(x^k) - t \left\| G_t^{i^*_{k+1}}(x^k) \right\|^2 = h(x^k) - t \max_{i=1,\ldots,m} \left\| G_t^i(x^k) \right\|^2. \tag{34}$$

(ii) From (i) we conclude that $h^k \equiv h(x^k)$ is a non increasing sequence, and since $h^* > -\infty$ it is bounded, therefore it has a limit which we will denote by $\tilde{h}$. Furthermore, since the sequence $\{x^k\}_{k\in\mathbb{N}}$ is bounded it has a converging subsequence $\{x^{k_q}\}_{q\in\mathbb{N}}$ which converges to $\tilde{x}$. Since h is lower semicontinuous the sequence $h(x^{k_q})$ must also converge to $\tilde{h}$. Therefore, $\tilde{h}$ is the value of h at any limit point of sequence $\{x^k\}_{k\in\mathbb{N}}$.

In order prove that any such limit point $\tilde{x}$ is stationary we assume to the contrary, that is $G_t(\tilde{x}) \neq 0$ (for convex g, a point is a stationary point if and only if it is a fixed point of the gradient step proximal mapping). Let $y = \tilde{x} + tG_t(\tilde{x})$ and $y^{k_q} = x^{k_q} + tG_t(x^{k_q}) \quad \forall q \in \mathbb{N}$. From non-expensiveness of the proximal operator for convex g we have that:

$$\|y - y^{k_q}\| \leq \|\tilde{x} - t\nabla f(\tilde{x}) - x^{k_q} + t\nabla f(x^{k_q})\| \leq (1 + tL)\|\tilde{x} - x^{k_q}\|,$$

where the last inequality is derived from the triangle inequality and the Lipschitz property of ∇f. Consequently, the convergence $\|\tilde{x} - x^{k_q}\| \to 0$ as $q \to \infty$ results in $\|y - y^{k_q}\| \to 0$. Moreover, the way $i^*_{k_q+1}$ is chosen, insures that

$$\|y^{k_q} - x^{k_q}\|^2 = t\|G_t(x^{k_q})\| \leq \sqrt{m}t\|G_t^{i^*_{k_q+1}}(x^{k_q})\| = \sqrt{m}\|x^{k_q+1} - x^{k_q}\| \leq \sqrt{h(x^{k_q}) - h(x^{k_q+1})}$$

and from convergence of h^k it follows that $\|y^{k_q} - x^{k_q}\| \to 0$ as $q \to \infty$. Finally,

$$\|y - x^{k_q}\| \leq \|y - y^{k_q}\| + \|y^{k_q} - x^{k_q}\|$$

so we conclude that $x^{k_q} \to y$ as $q \to \infty$ and therefore $y = \tilde{x}$ in contradiction to $\tilde{x}$ not being stationary.

(iii) The convergence rate for $\|G_t^{i_{k+1}^*}(x^k)\|$ is given similarly to Proposition 6.1 (ii) by summing the result of (i) from 0 to $N-1$ and bounding $h(x^N) \geq h(x^*)$. Using the fact that $\|G_t(x^k)\|^2 \leq m \max_{i=1,\ldots,m} \|G_t^i(x^k)\|^2 = m\|G_t^{i_{k+1}^*}(x^k)\|^2$ we get the convergence rate for G_t.

□

Choosing, for example $t = 1/L$ where $L \geq L_f$ we have that

$$\min_{k=0,\ldots,N-1} \left\| \text{prox}_h(x^k) - x^k \right\| \leq \sqrt{\frac{m^2}{N}}. \tag{35}$$

While convergence of both the CBPGM and GBPGM versions implies convergence of the global proximal step, the rate of convergence for both block versions can not be compared. This is due to the difference between the measures of convergence. The measure for the GBPGM can be directly translated to global proximal step convergence, while the measure for the CBPGM can not, and deals only with block proximal step. However, both versions have the same computational cost per iteration, since in each we compute m block-wise proximal gradients per iteration. Although the GBPGM only implements one of the m block steps it computes, parallelization can be utilized in order to achieve a faster effective rate (see [12]). Such sub-iteration parallelization is impossible for the CBPGM, since the sub-iterations are dependent. We can also conclude that, theoretically at least, taking the same step size t, the PGM converges faster than GBPGM with the same number of operations per iteration (both algorithms compute the proximal mapping at each block).

We add some remarks about the aforementioned proximal algorithms concerning problem (PN).

Remark 6.1. (i) As stated, these algorithms converge to a stationary point (or KKT point). Since, by definition, problem (PN) is regular (the gradient of active constraints are independent for any optimal solution), and the constraints are convex, this means that this point satisfies the necessary conditions for local optimum. Recall that since the problem is not convex a stationary point is not necessarily a local optimum.

(ii) Not every stationary point is "good", e.g., every point in the null-space of the matrix $\tilde{E}$ is a stationary point (its gradient is equal to zero) but its objective function value equals 0. Therefore, starting any proximal gradient algorithm from $x^0 = 0$ is obviously a bad choice,

since it will be the algorithm's solution although it is not a local maximum (assuming $\tilde{\mathcal{A}} \neq 0$). A good choice for x^0 might be the eigenvector of $\tilde{\mathcal{A}}$ associated with its largest eigenvalue. Alternatively $x^0 = [x_1^0, \ldots, x_m^0]$ where x_i^0 is the eigenvector of matrix $\tilde{A}_i^T \tilde{A}_i$ associated with its largest eigenvalue, and sign such that $\sum_{j<i} x_i^0 \tilde{A}_i^T \tilde{A}_j x_j^0 \geq 0$. Both options will guarantee that the initial solution is "far enough" from the null-space of $\tilde{E}$.

In section 7 we will present another method to obtain stationary points for problem (PN), which satisfy additional global optimality conditions. Therefore, this algorithm is more robust with regard to the starting point x^0, i.e. we can start at $x^0 = 0$ and obtain a solution with value larger then 0.

7 Block-wise Optimal Ascent

In this section, we attempt to devise a method which utilizes the block structure of problem (PN) in order to find the global maximum. We will do so by first constructing solvable block-wise sub problems, and then by using the solutions to these problem to generate a sequence of improving points for the entire problem.

For some fixed $x_{-i} \in C_{-i}$, we minimize the objective of problem (PN) with respect to x_i. Denoting the sub-problem parameters as:

$$\begin{aligned} &\tilde{\mathcal{A}}_i = \tilde{A}_i^T \tilde{A}_i, \quad c_j = \tilde{A}_j x_j \in \mathbb{R}^M, \quad c_{-i} = \sum_{\substack{j \in \{1,\ldots,n\}. \\ j \neq i}} c_j \\ &a_i = 2\tilde{A}_i^T c_{-i}, \quad d_i = \|c_{-i}\|^2, \end{aligned} \tag{36}$$

the block i sub-problem is:

$$\begin{aligned} \max_{x_i \in \mathbb{R}^{n_i}} \; & x_i^T \tilde{\mathcal{A}}_i x_i + a_i^T x_i + d_i \\ s.t. \quad & x_i^T x_i \leq 1 \end{aligned} \tag{P_i}$$

This problem is a particular case of the problem known as the *Trust Region Subproblem* we discussed in section 4. The problem was described by Sorenson in [44], as a subproblem of a general algorithm for solving non-convex problem. Later Moré and Sorenson [32] discussed, in depth, an algorithm to solve this sub-problem, which we will refer to as the *Trust Region Subproblem Algorithm* (TRSPA). The solution first requires diagonalization of matrix $\tilde{\mathcal{A}}_i$ by some orthonormal matrix U

(the eigenvector matrix). The problem becomes:

$$\begin{aligned} &\max_{z\in\mathbb{R}^{n_i}} \sum_{j=1}^{n_i} \lambda_j z_j^2 + 2{\xi_i}^T z + d_i \\ &s.t. \quad \|z\| \leq 1 \end{aligned} \tag{$P1_i$}$$

where $U\tilde{\mathcal{A}}_i U^T = \Lambda_i = Diag(\lambda_1, \ldots, \lambda_{n_i})$, $\xi_i = Ua_i/2$ and $z = Ux_i$. The solution scheme is based on finding the optimal value of the, strictly positive, dual scalar variable $\tilde{\lambda}$, which is the root of the function:

$$\psi(\tilde{\lambda}) = g^T(\tilde{\lambda}I - diag(\lambda))^{-2}g - \Delta^2 = \sum_{i=1}^{n} \frac{\xi_i^2}{(\tilde{\lambda} - \lambda_i)^2} - \Delta^2. \tag{37}$$

A procedure of finding $\tilde{\lambda}$ is presented by Sorenson and Moré [32] and described further in [37]. It is based on the Newton method safeguarded on the open segment $[\lambda_{max}, \infty)$ where $\lambda_{max} = \max_i \lambda_i$.

We can, therefore, conclude that the convergence rate is better than linear (bisection) and at most quadratic (Newton). The number of operations of each stage of our formulation is $O(n_i)$, and initializing through SVD takes $O(n_i^3)$. The SVD is less numerically stable for large dimensions, so if n_i is large we can use Cholesky decomposition at each stage at computational cost of $O(n_i^3)$.

We can now devise an algorithm to solve (PN) which stems from the solution to problem (P_i), starting from some point in the feasible set and optimize block by block, iteratively. To formalize the algorithm we use the same notations as in Section 6, i.e. using the minimum formulation, denoting $f(x)$ as the objective function and $f_i(x_i|x_{-i})$ the objective function of block i sub-problem (recall that both are concave). The formulation is then given in algorithm 2 which we will refer to as the *Block Optimal Ascent* (BOA) algorithm.

The BOA can be viewed as an algorithm for non cooperative game: each player i selects a strategy x_i which is a point on a sphere so as to maximize her utility function, which depends on the other players' strategies. Since we are talking about Pure Strategy Nash equilibrium we know that generally one doesn't necessarily exist. In this case, however, such an equilibrium must exist since the individual utility functions are identical to each other, and (equal the problem's objective function). The algorithm continues until no player has an improving strategy.

This method is a variation of the *Alternating Minimization Method* or *Gauss-Seidel* (GS)

Algorithm 2 Block-wise Optimal Ascent (BOA)

1: Initialize $k = 0$, $x^0 = 0 \in \mathbb{R}^n$, $\delta \gg 1$
2: Set $z^{k,0} \leftarrow x^k$
3: **for** $i = 1$ to m **do**
4: Solve (P_i) holding $z_{-i}^{k,i-1}$ constant and let $y \in \arg\min_x f_i(x|z_{-i}^{k,i-1}) \subseteq \mathbb{R}^{n_i}$ obtained by using the TRSPA.
5: **if** $f_i(y|z_{-i}^{k,i-1}) \geq f_i(z_i^{k,i-1}|z_{-i}^{k,i-1})$ **then**
6: $y \leftarrow z_i^{k,i-1}$
7: **end if**
8: Update $z_i^{k,i} \leftarrow y$ and $z_{-i}^{k,i} \leftarrow z_{-i}^{k,i-1}$
9: **end for**
10: Update $x^{k+1} \leftarrow z^{k,m}$
11: Update $\delta \leftarrow \min\{\|x^{k+1} - x^k\|, \delta\}$
12: **if** $\delta < \epsilon_1$ **then**
13: Stop algorithm and return x^{k+1}
14: **else**
15: Update $k \leftarrow k + 1$ and return to step 2.
16: **end if**

method presented in [12], which is based on successive global minimization with respect to each coordinate in a cyclic order. Linear convergence was proven for unconstrained problems where the objective function is strictly convex in each coordinate (see [12, 28]). Convergence was also proven for quasi-convex problems [45]. However, such algorithms may fail to converge, as illustrated in [40] for unconstrained minimization of a differentiable function. In [22] the authors apply the GS method to constrained minimization, where the domain of each block is a convex set, and show that under some assumptions, such as strict block-wise quasi-convexity, the method converges to a global optimum. They also prove, that in a general setting, any limit point is a stationary point for each of the blocks. All these results assume a cyclic version of the alternating method, and usually some form of convexity of the function to be minimized.

We will now show some properties of this algorithm as applied to problem (PN), and derive convergence results for this *non-convex* case. We refer to solving sub-problem (P_i) for block i as sub-iteration i, and to all m sub-iterations as an iteration.

Theorem 7.1. *Starting from a feasible x^0 such that $x_i^0 \in null(\tilde{A}_i)$, after one iteration ($m$ sub-*

iterations), the BOA algorithm reaches a solution y such that

$$y^T \tilde{\mathcal{A}} y \geq \sum_{i=1}^{m} \lambda_{max}(\tilde{\mathcal{A}}_i) = \sum_{i=1}^{m} \left\| \tilde{A}_i \right\|^2$$

.

Proof. Notice that starting with x^0, after sub iteration i we obtain $z^{0,i} = (x_1^1, \ldots, x_i^1, x_{i+1}^0, \ldots, x_m^0)$, where $y = x^1$. At each sub-iteration we can choose $\pm v_{\max}^i$ where $v_{\max}^i$ is the eigenvector of $\tilde{\mathcal{A}}_i$ corresponding to its maximal eigenvalue. Furthermore, summing over the two equations defining $f_i(\pm v_{\max}^i | z_{-i}^{0,i})$ we come to:

$$-\frac{f_i(v_{\max}^i | z_{-i}^{0,i}) + f_i(-v_{\max}^i | z_{-i}^{0,i})}{2} = \left(\left\| \tilde{A}_i \right\|^2 + \left\| \tilde{A}_{-i} z_{-i}^{0,i} \right\|^2 \right) = \left(\left\| \tilde{A}_i \right\|^2 + \left\| \sum_{j=1}^{i-1} \tilde{A}_j x_j^1 \right\|^2 \right)$$

Since x_i^1 is the optimal value in sub-iteration i we have that: $f_i^* = f_i(x_i^1 | z_{-i}^{0,i}) \leq f_i(\pm v_{\max}^i | z_{-i}^{0,i})$ and so

$$f_i(x_i^1 | z_{-i}^{0,i}) = -\left\| \sum_{j=1}^{i} A_j x_j^1 \right\|^2 \leq -\left\| \tilde{A}_i \right\|^2 - \left\| \sum_{j=1}^{i-1} A_j x_j^1 \right\|^2 = -\|\tilde{A}_i\|^2 + f_{i-1}\left(x_{i-1}^1 | z_{-(i-1)}^{0,i-1} \right).$$

Since $f(x^0) = 0$ we can now show by induction that $f_i(x_i^1 | z_{-i}^{0,i}) \leq -\sum_{j=1}^{i} \left\| \tilde{A}_j \right\|^2$ for all $i = 1, \ldots, m$. Therefore, after m sub-iteration we reach the desired result. □

Definition 7.1. A vector x is called a global block-wise optimal solution with respect to problem (PN) if:

$$f_i(x_i | x_{-i}) \leq f_i(y | x_{-i}) \; \forall y \in C_i, \; i = 1, \ldots, m$$

i.e. x_i is globally optimal for problem (P_i) given x_{-i}.

Assumption 7.1. Without loss of generality we will assume that $\|\tilde{A}_i\|_2 > 0$. Indeed otherwise $\tilde{A}_i = 0$ and the block can be eliminated.

Lemma 7.2. *Given problem* (PN)*, which satisfies assumption 7.1, let x be a point for which all the constraints are active. Then,*

(i) If x is a global block-wise optimal solution then it satisfies the KKT first order necessary conditions for local optimum*.*

(ii) Conversely, if x is globally optimal then it is also a global block-wise optimal.

(iii) If x is a global block-wise optimal solution it either satisfies the second order necessary conditions for local maximum or an improving point can be found efficiently.

Proof. The proof is given in appendix C □

Theorem 7.3. *Consider problem* (PN) *and let assumption 7.1 hold true. Let $\{x^k\}_{k\in\mathbb{N}}$ be the sequence of points generated by the BOA algorithm starting from some feasable solution x^0. The following assertions hold:*

(i) The sequence of objective function values $\{f(x^k)\}_{k\in\mathbb{N}}$ is non increasing and converges to some value f^*. Furthermore, $f(x^{k+1}) = f(x^k)$ iff $x^{k+1} = x^k$.

(ii) Each point x^k, $k > 0$ in the sequence satisfies that $\|x_i^k\| = 1$, $i = 1, \ldots, m$

(iii) Any converging subsequence of $\{x^k\}_{k\in\mathbb{N}}$ converges to some stationary point $x^\dagger$, which is also a block-wise optimal point, such that $f(x^\dagger) = f^*$.

Proof. (i) At each sub iteration i:

$$f(z^{k,i}) = f_i(z_i^{k,i}|z_{-i}^{k,i}) = f_i(z_i^{k,i}|z_{-i}^{k,i-1}) \leq f_i(z_i^{k,i-1}|z_{-i}^{k,i-1}) = f(z^{k,i-1})$$

and so $f(x^k) = f(z^{k,0}) \geq f(z^{k,m}) = f(x^{k+1})$. Since f is obviously bounded below (the domain is compact) the value of f converges to some f^*. In other words, $\forall\epsilon > 0$, $\exists N_\epsilon : f(x^k) - f^* \leq \epsilon \quad \forall k > N_\epsilon$. If $f(x^{k+1}) = f(x^k)$ we have that $f_i(z_i^{k,i}|z_{-i}^{k,i}) = f_{i-1}(z_i^{k,i-1}|z_{-i}^{k,i})$ and by algorithm definition $z_i^{k,i} = z_i^{k,i-1}$, $i = 1, \ldots, m$ so that $x^{k+1} = x^k$.

(ii) In the proof of Theorem 7.1 we showed that for each iteration k and sub-iteration i we get that $z_i^{k,i}\tilde{A}_i\tilde{A}z^{k,i} \geq \|A_i\|_2^2 > 0$. If $0 \neq \|z_i^{k,i}\| < 1$ dividing $z_i^{k,i}$ by its norm will improve the objective function value. Therefore, the optimal solution for each sub-iteration is on the boundary of

the corresponding block. If we start from a non-extreme point, we obtain an extreme point after at most m sub-iterations. We can now conclude that $\|x_i^k\| = 1,\ i = 1, \ldots, m,\ \forall k > 0$,

(iii) Since the domain is compact, the sequence $\{x^k\}_{k\in\mathbb{N}}$ must have a converging subsequence $\{x^{k_l}\}_{l\in\mathbb{N}}$ which converges to some point $x^\dagger$. Since $f(x^k)$ converges to f^* and the function f is continuous, it follows that $f(x^\dagger) = f^*$.

We will now show that such $x^\dagger$ is indeed global block-wise optimal. We will assume to the contrary, that $x^\dagger$ is not global block-wise optimal, meaning that $\exists i, y_i : f_i(y_i|x^\dagger_{-i}) < f_i(x^\dagger_i|x^\dagger_{-i}) = f^*$. Without loss of generality we assume $i = m$, since we can define the series x^k as the results of sub-iteration i. Let us denote $\epsilon = f^* - f_m(y_m|x^\dagger_{-m})$. From the convergence of x^{k_l} we know that for some l we have $\|\overline{x}^{k_l} - \overline{x}^\dagger\| \leq \delta$ and specifically $\|x^{k_l}_{-m} - x^\dagger_{-m}\| \leq \delta$. f is continuous, therefore, for a small enough δ we obtain:

$$\begin{aligned} \left|f_m(y_m|x^{k_l}_{-m}) - f_m(y_m|x^\dagger_{-m})\right| &< \epsilon/2 \\ \left|f_m(x^{k_l}_m|x^{k_l}_{-m}) - f_m(x^\dagger_m|x^\dagger_{-m})\right| = \left|f_m(x^{k_l}_m|x^{k_l}_{-m}) - f^*)\right| &< \epsilon/2 \end{aligned} \tag{38}$$

but we know that $f_m(y_m|x^{k_l}_{-m}) \geq f_m(x^{k_l}_m|x^{k_l}_{-m}) \geq f_m(x^\dagger_m|x^\dagger_{-m}) = f^*$ from optimality of $x^{k_l}_m$. Summing over these inequalities we get:

$$\begin{aligned} f_m(y_m|x^\dagger_{-m}) - f^* &= f_m(y_m|x^\dagger_{-m}) - f_m(y_m|x^{k_l}_{-m}) + f_m(y_m|x^{k_l}_{-m}) - f^* \\ &\leq f_m(x^\dagger_m|x^\dagger_{-m}) - f_m(x^{k_l}_m|x^{k_l}_{-m}) + f_m(y_m|x^{k_l}_{-m}) - f^* < \epsilon \end{aligned}$$

which results in a contradiction.

The fact that x^* is a critical point is then readily given by (ii) and Lemma 7.2.

□

Given an optimal solution x^*, $Opt(PN) = \left\|\sum_{i=1}^m \tilde{A}_i x_i^*\right\|^2 \leq m\sum_{i=1}^m \|A_i\|^2$ holds true. Combining this fact with theorem 7.1 we obtain the following result.

Corollary 7.4. *Starting from some feasible point x^0 such that $x_i^0 \in null(\tilde{A}_i)$, and denoting the limit of the sequence of the objective function values generated by BOA as $Val(BOA)$, we obtain $Opt(PN) \geq Val(BOA) \geq \frac{1}{m} Opt(PN)$.*

7.1 Rate of Convergence of BOA

The analysis of the rate of convergence for the cyclic BOA algorithm is not straight forward. Instead we use a greedy version, which is similar in concept to the *Greedy Block Proximal Gradient Method* (GBPGM) presented in Section 6. We will refer to this greedy variant as *Greedy Block Optimal Ascent* (GBOA) and to the cyclic version as *Cyclic Block Optimal Ascent* (CBOA). Notice that the proofs of Theorem 7.1 can be extended to GBOA, and initializing GBOA by a cyclic iteration will insure that Theorem 7.3 still holds true (where a non ascending sequence is generated even without such initialization). So we will assume, as in CBOA, that GBOA generates a sequence of extreme points of C.

We will define two operators $P_t^i(x) \equiv tG_t^i(x) = \text{prox}_{tg_i}(x_i - t\nabla f_{x_i}(x)) \in \mathbb{R}^{n_i}$ and $T^i(x) \in \arg\min_{x_i} f_i(x_i|x_{-i})$ and subsequently:

$$\tilde{y}_j^{k+1,i} = \begin{cases} P_t^i(x^k) & j = i \\ x_j^k & j \neq i \end{cases}, \quad \hat{y}_j^{k+1,i} = \begin{cases} T^i(x^k) & j = i \\ x_j^k & j \neq i \end{cases}.$$

We also define $i_{k+1}^* = \arg\min_{i=1,\ldots,m} f(y^{k+1,i})$ and $x^{k+1} = \hat{y}^{k+1,i_{k+1}^*}$. Using the same notations for h, G and $t > 0$ as in GBPG and combining the proof to Theorem 6.2 and the optimality of $\hat{x}_i^{k+1}$ and i_{k+1}^* we have that:

$$h(x^{k+1}) \leq h(\hat{y}^{k+1,i}) \leq h(\tilde{y}^{k+1,i}) \leq h(x^k) - t\|G_t^i(x^k)\|^2 \tag{39}$$

and similarly to GBPG the following rate of convergence is achieved:

$$\min_{k=0,\ldots,N-1} \|G_t(x^k)\| \leq \sqrt{\frac{m(h(x^0) - h^*)}{tN}}, \; t > 0 \tag{40}$$

The problem with this result is that it just shows that we reach some stationary point, but it doesn't show the strength of BOA, which guarantees that if that stationary point is not block wise optimal then the algorithm continues to run and obtains a better objective function value.

In order to show the convergence to a block-wise global optimum with the same rate we will look at the problem a bit differently. We define the sets $Lev_i(x) = Lev_{f(x)}(f_i) = \{y_i : f_i(y_i|x_{-i}) \leq f(x)\}$

which are the sublevel sets of f_i in the appropriate subspaces. We also define the sets of all feasible vectors with objective not worse than $f(x)$ and which differ from x only on the ith block as:

$$X_i(x) = \left\{ y : y_i \in Lev_i(x) \bigcap C_i,\ y_j = x_j,\ \forall j \neq i \in \{1, \dots, m\} \right\}.$$

We will aim to show that for each point in these sets applying the proximal gradient causes a bounded and decreasing step size and so the method converges to a point which is not only stationary but can not be improved at any block (a block-wise global optimum). We will first notice the following interesting property:

Lemma 7.5. *Let x be a feasible point of problem (PN) which satisfies:*

$$\|P_t^i(y) - y\| = 0,\ \forall y \in X_i(x),\ i = 1, \dots, m$$

then x is block-wise global optimal point of problem (PN) .

Proof. Let us assume to the contrary that x is not a block-wise global maximal point than $\exists i$ and a point $y \in X_i(x)$ such that $f(y) < f(x)$. We can assume, without loss of generality, that y_i is in the interior of C_i, since f is continuous. Therefore, $P_t^i(y) = 0$ if and only if $\nabla f_i(y_i|x_{-i}) = 0$ but since f_i is a concave function according to the gradient inequality we have that:

$$f(x) \leq f(y) + (x - y)^T \nabla f(y) = f(y) + (x_i - y_i)^T \nabla f_i(y_i|x_{-i}) = f(y)$$

in contradiction to the assumption that $f(y) < f(x)$. □

We can therefore conclude that if all the limits points of GBOA satisfy the above, they are all block-wise globally optimal, which leads us to the following.

Theorem 7.6. *Let $\{x^k\}_{k\in\mathbb{N}}$ be the sequence generated by GBOA, starting from point x^0. Any limit point of the sequence is a block-wise global optimum and the rate of convergence is given by*

$$\min_{k=1,\dots,N} \sum_{i=1}^{m} \max_{y \in X_i(x^k)} \|G_t^i(y)\| \leq m \sqrt{\frac{h^* - h(x^0)}{tN}}, \quad \forall t > 0.$$

Proof. By the monotonicity of $h(x)$ over the sequence generated by the proximal gradient step, shown in Theorem 6.2, and optimality of i^*_{k+1} we have that equation (39) can be written for any $y \in X_i(x^k)$ as:

$$t\|G^i_t(y)\|^2 \leq h(y) - h(P^i_t(y)) \leq h(y) - h(x^{k+1}) \leq h(x^k) - h(x^{k+1}) \quad \forall y \in X_i(x^k),\ i = 1, \ldots, m.$$

Therefore,

$$\max_{i=1,\ldots,m} \max_{y \in X_i(x^k)} t\|G^i_t(y)\|^2 \leq h(x^k) - h(x^{k+1}) \tag{41}$$

Summing over all $k = 0, \ldots, N-1$ we get that:

$$\sum_{k=0}^{N-1} \left\{ \max_{i=1,\ldots,m} \max_{y \in X_i(x^k)} t\|G^i_t(y)\|^2 \right\} \leq h(x^0) - h(x^N) \leq h(x^0) - h^* \tag{42}$$

We can conclude that

$$\begin{aligned} \min_{k=1,\ldots,N} \sum_{i=1}^{m} \max_{y \in X_i(x^k)} \|G^i_t(y)\|^2 &\leq \min_{k=1,\ldots,N} \left\{ m \max_{i=1,\ldots,m} \max_{y \in X_i(x^k)} \|G^i_t(y)\|^2 \right\} \\ &\leq \frac{m(h(x^0) - h^*)}{tN} \end{aligned} \tag{43}$$

Or alternatively

$$\min_{k=1,\ldots,N} \sum_{i=1}^{m} \max_{x^\dagger \in X_i(x^k)} \|G^i_t(x^\dagger)\| \leq m\sqrt{\frac{h(x^0) - h^*}{tN}} \tag{44}$$

□

Computing this convergence measure might be impossible but it does provide an upper bound for the number of iterations needed to obtain a certain accuracy. Furthermore, notice that there is an added factor of $\sqrt{m}$ to the rate of convergence given in equation (40), which proves stationarity alone.

A potential problem with implementing this method is that at every stage there is a need to compute the optimal value for each of the blocks. This additional computational cost can be dealt with by parallelization, which allows to compute all or some of these simultaneously without increasing run-time, as suggested in [12].

7.2 BOA for the MAX-CUT problem

The Max-Cut is a particular instance of our general problem, where all the block elements are singletons. For this problem the BOA is actually a local search scheme which is known to be a $\frac{1}{2}$-approximation of the problem [42] (in contrast to the $\frac{1}{n}$ approximation we have proven here). This is not a good approximation, since it can also be obtained by a naive randomized cut method. This local search also admits a local optimum point, in the continuous sense, since the feasible set is an n dimensional box and we are at a vertex which is better than all adjacent vertices, hence, by convexity of the objective function, we are in a local maximum. Polynomial time convergence of $O(n^2)$ [39] is known for the non-weighted case and cubic graphs (graphs where each vertex has a maximal degree of 3) and $O(n^2 \sum_{i,j} |w_{ij}|)$ when weights are polynomially bounded, although the problem is generally (*PLS-complete*) (similar to NP-complete with respect to local optimum) as was proven in [16]. We showed that the number of iteration needed to obtain an accuracy of ϵ is $O(L_f \frac{n^2}{t\epsilon^2})$ (sub-linear convergence), where $L_f = \|W\|$ where W is the Laplacian matrix (or some PSD variation of the matrix) and n is the number of vertices.

7.3 Randomized BOA

Greedy BOA forces us to solve all m block problems in each iteration. Although this procedure can be parallelized when m is large, it may cause high computational overhead. Randomizing the choice of the current block, can potentially reduce the number of operations needed to obtain the same convergence accuracy. Randomization here means that at each iteration k we pick at random index i_k of the block to be optimized, from the set $J_k = \{1, \ldots, m\}/\{i_{k-1}\}$. We call this scheme *Randomized BOA* (RBOA).

The following proof shows that all limits points of RBOA are block-wise optimal and that its rate of convergence is similar to GBOA.

Theorem 7.7. *Any limit point of the sequence the sequence* $\{x^k\}_{k\in\mathbb{N}}$ *generated by RBOA, starting*

from point x^0, is a block-wise global optimum. Its rate of convergence is given by

$$\mathbb{E}\left(\min_{k=0,\dots,N-1}\sum_{i=1,\dots,m}\max_{x^\dagger\in X_i(x^k)}\|G_t^i(x^\dagger)\|\right)\leq m\sqrt{\frac{(m-1)(h(x^0)-h^*)}{tN}},\quad \forall t>0.$$

The idea behind the proof is that on average every $m-1$ iteration we choose the partition element which gives us the best objective function value (as in the GBOA), and so we need to increase the number of iterations by a factor of $m-1$ to get the same average convergence rate.

Proof. Given the sequence $\{x^k\}_{k\in\mathbb{N}}$ we define the random variables

$$\varphi^k=\max_{i=1,\dots,m}\max_{y\in X_i(x^k)}\|G_t^i(y)\|^2,\quad \theta^k=\sum_{i=1}^m\max_{y\in X_i(x^k)}\|G_t^i(y)\|,\quad \forall t>0.$$

We will also denote by i_k^* the index that the GBOA algorithm choses at the given point x^k, and let i_k be the index of the block chosen randomly (uniformally) from J_k by RBOA at iteration k.

If $i_k\neq i_k^*$ then: $0\leq t\|G_t^i(x^k)\|^2\leq h(x^k)-h(x^{k+1})$ otherwise, with probability of at least $\frac{1}{m-1}$ we pick i_k^*, and by Theorem 7.6: $t\varphi^k\leq h(x^k)-h(x^{k+1})$. Taking the expectation over all possible realizations of i_j $j=1,\dots,k$ we obtain that:

$$\begin{aligned}\mathbb{E}\left(h(x^k-h(x^{k+1}))\right)&=\mathbb{E}\left(\mathbb{E}(h(x^k)-h(x^{k+1})|x^k)\right)\\&\geq\mathbb{E}\left(P(i_k\neq i_k^*)*0+P(i_k=i_k^*)(t\varphi^k)\right)=\mathbb{E}\left(\frac{1}{m-1}t\varphi^k\right)\end{aligned}\tag{45}$$

Summing over iterations $1,\dots,N-1$ we obtain the following.

$$\begin{aligned}\sum_{k=0}^{N-1}\mathbb{E}\left(\varphi^k\right)&\leq\frac{m-1}{t}\sum_{k=0}^{N-1}\mathbb{E}\left(f(x^{k+1})-f(x^k)\right)\\&=\frac{m-1}{t}\mathbb{E}\left(f(x^N)-f(x^0)\right))\leq\frac{m-1}{t}(f^*-f(x^0))\end{aligned}\tag{46}$$

Using the fact that $\theta^k\leq(m\sqrt{\varphi^k})$ and Jensen inequality for the square root function $\mathbb{E}(\sqrt{x})\leq$

$\sqrt{\mathbb{E}(x)}$ leads to the following:

$$\begin{aligned}\mathbb{E}\left(\min_{k=0,\ldots,N-1}\theta^k\right) &\leq m\mathbb{E}\left(\min_{k=0,\ldots,N-1}\sqrt{\varphi^k}\right) = m\mathbb{E}\left(\sqrt{\min_{k=0,\ldots,N-1}\varphi^k}\right)\\ &\leq m\sqrt{\mathbb{E}\left(\min_{k=0,\ldots,N-1}\varphi^k\right)} \leq m\sqrt{\frac{1}{N}\sum_{k=0}^{N-1}\mathbb{E}(\varphi^k)}\\ &\leq \sqrt{\frac{(m-1)m^2(h(x^0)-h^*)}{tN}}\end{aligned}$$

as claimed in the theorem. □

Comparing the convergence rate of GBOA and RBOA, one may observe that for a given accuracy, the number of iterations needed by th RBOA is larger by a factor of $\sqrt{m}$. This is misleading, since the number of block sub-iterations needed actually decreases by a factor of $\sqrt{m}$, so the actual expected computational cost remains the same.

Notice that since the random variable θ^k is non negative the convergence of the expectation implies, by Markov inequality, a convergence in probability, since

$$P\left(\min_{k=0,\ldots,N-1}\theta^k > \epsilon\right) \leq \frac{\mathbb{E}\left(\min_{k=0,\ldots,N-1}\theta^k\right)}{\epsilon} = \frac{1}{\epsilon}\sqrt{\frac{m^3L(f^*-f(x^0))}{N}} \underset{N\to\infty}{\longrightarrow} 0$$

So we can conclude that if $\sqrt{\frac{m^3L(f^*-f(x^0))}{N}} = \epsilon\delta$, then the probability of $\min_{k=0,\ldots,N-1}\theta^k \leq \delta$ is grater than $1-\epsilon$, and so the number of iteration which insure that are $N_{\epsilon,\delta} = \frac{m^3L(f^*-f(x^0))}{\epsilon^2\delta^2}$.

The downside of the algorithm is that it does not have a stopping criteria, since any such criteria is dependent on all the partition elements, and so checking the validity of the criteria will be exactly like solving the greedy BOA. We can therefore choose one of the following courses of action: Checking the criteria every constant number of iterations K, thus increasing the computational cost by the equivalent of additional $\frac{mN}{K}$ iteration; The alternative is to run the algorithm for $N_{\epsilon,\delta}$ iterations given choice of ϵ and δ without checking the criteria validity; The third option is to check objective function convergence instead.

The algorithm can be applied even more efficiently if at each point we remember which block

elements were already tried and can not be improved, thus eliminating repetitions, and stopping automatically when there are no more improving blocks.

8 Lagrangian Guided Ascent

We showed that the approximation resulting from the BOA algorithm is affected by its starting point. Therefore, we will try to find a "good" starting point, insuring a better approximation. This will be done by utilizing the information received from the SDP solution.

The dual of problem (PN) is given by:

$$\begin{aligned} &\min_{\mu\in\Re^m_+} \sum_{i=1}^m \mu_i \\ &s.t. \\ &\quad \sum_{i=1}^m \mu_i \mathcal{I}_i - \tilde{\mathcal{A}} \succeq 0. \end{aligned} \tag{DPN}$$

The SDP relaxation is in fact the dual of DPN. By weak duality, and regularity of SDP, we have

$$Opt(PN) \leq Opt(DPN) = Opt(SDP). \tag{47}$$

Assume $Opt(PN) = Opt(SDP)$ and $\mathcal{X}^*$ an optimal SDP solution for which all the constraints are active. Let $\mu^* \in \Re^m_{++}$ be the unique optimal dual variable vector, and x^* be a global maximizer of (PN), then by first order optimality conditions the following holds true.

$$\left(\tilde{\mathcal{A}} - \sum_{i=1}^m \mu_i^* \mathcal{I}_i\right) x^* = 0 \tag{48}$$

We build our starting point based on this fact. For convenience we will denote $P = \tilde{\mathcal{A}} - \sum_{i=1}^m \mu_i^* \mathcal{I}_i$.

Let $\{u_j\}_{j=1}^p$ be the eigenvectors of $\mathcal{X}^*$ such that $\{u_j\}_{j=1}^p \subset Null(P)$. By complementarity of the SDP we have that all the eigenvectors corresponding with the non-zero eigenvalues, denoted by $\{u_j\}_{j=1}^r$, are part of this group, therefore $p \geq r$, where $r = rank(\mathcal{X}^*)$. The set $\{u_j\}_{j=1}^p$ forms an orthonormal basis of the null space of P, therefore, x^* is a linear combination of these vectors. If $r = 1$ it is clear that the optimal solution is equal to $\sqrt{\lambda_1}u_1$ where λ_1 is the corresponding eigenvalue.

In general, of course, $r > 1$, so we devise a procedure, called *Lagrange Guided Ascent* (LGA), which will obtain a solution satisfying the first order necessary conditions with respect to μ^*. This procedure is presented in Algorithm 3. In our notations, we use the minimization formulation with concave objective function f. Let $C(\beta) = \{x : \|x_i\|^2 \leq \beta,\ i = 1, \ldots, m\}$ and let f^* be the SDP optimal solution.

Algorithm 3 Lagrangian Guided Ascent (LGA)

1: initialize $k = 0,\ y^0 = 0, x^0 = y^0\ s^0 = 0$.
2: Update $s^{k+1} \leftarrow \frac{1}{2}(1 + s_k)$.
3: Calculate $\rho_j = \arg\min_{\hat{\rho}} \left\{ f(y^k + \hat{\rho} u_j) : y^k + \hat{\rho} u_j \in C(s^{k+1}) \right\},\ j = 1, \ldots, m$
4: Find $j^* \in \arg\min_{j=1,\ldots,p} f(y^k + \rho_j u_j)$.
5: Update $y^{k+1} \leftarrow y^k + \rho_{j^*} \overline{u}_{j^*}$ and $x^{k+1} = x^{y+1}/\sqrt{s_{k+1}}$.
6: **if** $(1 - \epsilon) f^* < f(x^{k+1}) \leq f(x^k)$ and $s^{k+1} < 1 - \delta$ **then**
7: Update $k \leftarrow k + 1$.
8: Goto step 2
9: **else**
10: Exit and return x^k and $f(x^k)$
11: **end if**

It easy to see that at each iteration k the scalar s_k is simply the sum of a geometrical series with ratio of 0.5, and so we have that $s_k = 1 - 0.5^k$. The stopping criteria, therefore, is met in at most $N = \log\left(\frac{1}{\delta}\right)$ iterations. Furthermore, it is clear that the generated sequence, $\{x^k\}_{k \in \mathbb{N}}$ is feasible, and that at least one constraint is satisfied with an equality. Moreover, the sequence $\{f(x^k)\}_{k \in \mathbb{N}}$ is non increasing and therefore has a limit. Also at each stage we pick the best vector, hence $\lim_{k \to \infty} f(x^k) \leq f(x)$ for any feasible x of the form $x = \alpha u_j$ for some $j \in \{1, \ldots, p\}$, and specifically for any $j \in \{1, \ldots, r\}$.

To prove the applicability of this procedure, we will show that for any j the optimization problem $\min_{\hat{\rho}} \left\{ f(y^k + \hat{\rho} u_j) : y^k + \hat{\rho} u_j \in C(s^{k+1}) \right\}$ is easily solvable. Indeed, since each constraint i is a convex quadratic function of $\hat{\rho}$, and 0 is a strictly feasible solution, then each constraint i produces a feasibility interval $[\underline{\rho}^i_j, \overline{\rho}^i_j]$ around 0, where the endpoints are the roots of the quadratic function $\|I_i(y^k + \hat{\rho} u_j)\|^2 - s^{k+1} = 0$. The feasible set is then the intersections of these intervals, i.e.

the interval $[\max_i \underline{\rho}_j^i, \min_i \overline{\rho}_j^i]$. Since $\tilde{\theta}_j(\hat{\rho}) = f(y^k + \hat{\rho}u_j)$ is a concave quadratic function defined over this interval, its minimal value is obtained in one of the interval's endpoints. Therefore, this problem can be solved by finding these endpoints and checking their objective value.

Notice, that the quality of this solution depends on the choice of the basis $\{u_j\}_{j=1}^p$. From the specific choice of $\{u_j\}_{j=1}^p$ as the eigenvectors of $\mathcal{X}^*$, and by the properties we listed above it follows that $Opt(SDP) \leq rVal(LGA)$, the proof is similarly to that of proposition 5.1. Consequently we arrive at the following conclusion.

Corollary 8.1. *Let $\{x_k\}_{k\in\mathbb{N}}$ be the sequence generated by LGA and let $Val(LGA) = \lim_{k\to\infty} f(x^k)$, then $Val(LGA) \leq Opt(PN) \leq rVal(LGA)$, and $r(r+1) \leq 2m$.*

An important remark about this procedure:

Remark 8.1. In order to apply LGA, we need to have both the primal and dual SDP solutions and to obtain the eigenvectors. This can be done in polynomial time, as mentioned earlier.

9 A Computational Study

Table 1 summarizes the algorithms we presented for solving problem (PN). In this table the acronyms LB and UB represent lower and upper bound respectively, Limits represents the properties of the algorithm's limits points, Rate - the algorithm's iteration count for a given tolerance ϵ and ICC is an acronym for iteration computational complexity, i.e. number of operations per algorithm iteration. To understand and compare the performance of the various methods we conducted several numerical experiments.

In order to compare the algorithms in Table 1 fairly we applied the same stopping criteria for all of them. This criteria is given by: $\min_k |f(x^k) - f(x^{k-1})| < mL_f\epsilon$ where x^k is the result of iteration k, if at each iteration we change only one block, and of one sub iteration otherwise, L_f is the Lipschitz constant of the gradient of f and so mL_f is an upper bound on objective function value.

All the experiments were conducted on a 64Bit Intel Core 2Duo at 2.66GHz with 8GB RAM. The code was ran using MATLAB v2012a and, unless otherwise specifies, the optimization problems

were solved using MOSEK v6.0.

Table 1: Summary of tested algorithms

Algorithm	**Bound**	**Description**	**Approximation**	**Convergence**	
SDP using $HRVW - alg$	UB	Relaxation with value $S^*(\mathcal{A})$	General $\frac{2}{\pi} - \left(1 - \frac{2}{\pi}\right) \frac{S^*(-\mathcal{A})}{S^*(\mathcal{A})}$	Limit	Matrix $\mathcal{X} \succcurlyeq 0$
			$m \leq 2$ 1 MAX-CUT 0.878 $\mathcal{A} \succeq 0$ $2/\pi$ $rank(\mathcal{A}) = r$ $1/r$	Rate	$O\left(\sqrt{n}\log(1/\epsilon)\right)$
				ICC	$O\left(n^3\right)$
PGM	LB	Proximal gradient method.	-	Limit	Feasible stationary point.
				Rate	$O\left(1/\epsilon^2\right)$
				ICC	$O(n^2)$
GBPGM	LB	Block proximal gradient method. The block is chosen according to a greedy criteria.	-	Limit	Feasible stationary point.
				Rate	$O\left(\sqrt{m}/\epsilon^2\right)$
				ICC	$O(n^2)$
CBOA	LB	Alternating block optimization. Cyclic order.	$\frac{1}{m}$.	Limit	Feasible, block optimal point.
				Rate	-
				ICC	$O\left(\sum_i n_i^3\right)$
GBOA	LB	Alternating block optimization. The block is chosen according to a greedy criteria.	$\frac{1}{m}$	Limit	Feasible, block optimal point.
				Rate	$O\left(m/\epsilon^2\right)$
				ICC	$O\left(\sum_i n_i^3\right)$
RBOA	LB	Alternating block optimization. The block is chosen at random.	$\frac{1}{m}$	Limit	Feasible, block optimal point.
				Rate	$O\left(m/\epsilon^2\right)$
				ICC	$O\left(\max_i n_i^3\right)$
LGA	LB	Initial point based on SDP relaxation solution. Alternating block optimization-CBOA.	$\frac{1}{\sqrt{m}}$	Limit	Same as CBOA.
				Rate	
				ICC	

Experiment 1 focuses on the MAX-CUT problem. The experiment is based on benchmarks specified in [30] which contains three experiment sets taken from several articles, and from the 7th DIMACS Implementation Challenge.

The benchmark [30] compares three state of the art algorithms for the MAX-CUT problem:SS, $CirCut$ and $VNSPR$. We compared these algorithms' performance to that of the four BOA type algorithms: Cyclic-BOA (CBOA), Greedy-BOA (GBOA), Randomized-BOA (RBOA) and Lagrangian Guided Ascent (LGA). The starting point for each of the first three algorithms was the zero vector. The LGA is used to generate the starting point after which it continues with the Cyclic-BOA algorithms (this is actually a BOA method with a "smarter" starting point). All the BOA algorithms were ran until the stopping criteria with $\epsilon = 10^{-8}$.

Since some of the benchmark problems did not have a known upper bound we computed an upper bound by the SDP relaxation solved by our specifically designed code which modifies the $HRVW$ algorithm adapted from [23]. Some of the problems had a non PSD Laplacian matrix (since not all weights in the MAX-CUT problem were non-negative), so in order to implement the various methods these matrices were transformed to PSD matrices, by adding a large enough constant to the matrix diagonal. This adjustment impacts the optimal value (by a known constant) but not the optimal solution. For such problems the SDP approximation is lower than $2/\pi$ (see Table 1). For PSD matrices with negative weights the SDP gives only a $2/\pi$ approximation and not 0.878 approximation of Goemans and Williamson.

In order to compare the performance of the algorithms we used the *Lower to Upper Bound Ratio (LBR)*,the ratio between the results obtained by the various methods and the SDP relaxation. This ratio is denoted by $LBR_{Alg} = Val(Alg)/Opt(SDP)$ where Alg is the compared algorithm's name, or "Best" if the SDP is compared to the best known solution. $Val(Alg)$ is the value the algorithm attains. This measure estimates the tightness of both Alg and SDP, such that $(1 - LBR_{Alg})$ is the percent they may deviate from the actual optimal solution. We obtained that the $LBR_{Best} \in [80\%, 100\%]$ for the original problems while the transformed problem (with corrected matrices) had $LBR_{Best} \in [93\%, 100\%]$ which is always better the lower bound obtained by the SDP. The difference between these results occurs due to adding a very large constant to the objective function, which

influences the ratio.

Since the best algorithm was not necessarily close to the SDP solution, we looked at an alternative performance measure. *Best Ratio (BR)* is the ratio between the algorithms solution and the best known solution, which we denote by:$BR_{Alg} = Val(Alg)/Val(Best)$. We then compare the algorithms using the empirical CDF (Cumulative Distribution Function) of this measure. The CDF is given by

$$CDF_{Msr}(x) = \frac{\sum_{i=1}^{|Set|} Msr_{Alg}(i) \leq x}{|Set|}$$

where $Msr_{Alg}(i)$ denotes the Msr measure value for algorithm Alg in instance i and $|Set|$ is the size of the experiment set. The results of this comparison are given in Figure 1. The lower and more to the right the graph is, the better the algorithm is, as it gets higher BR values.

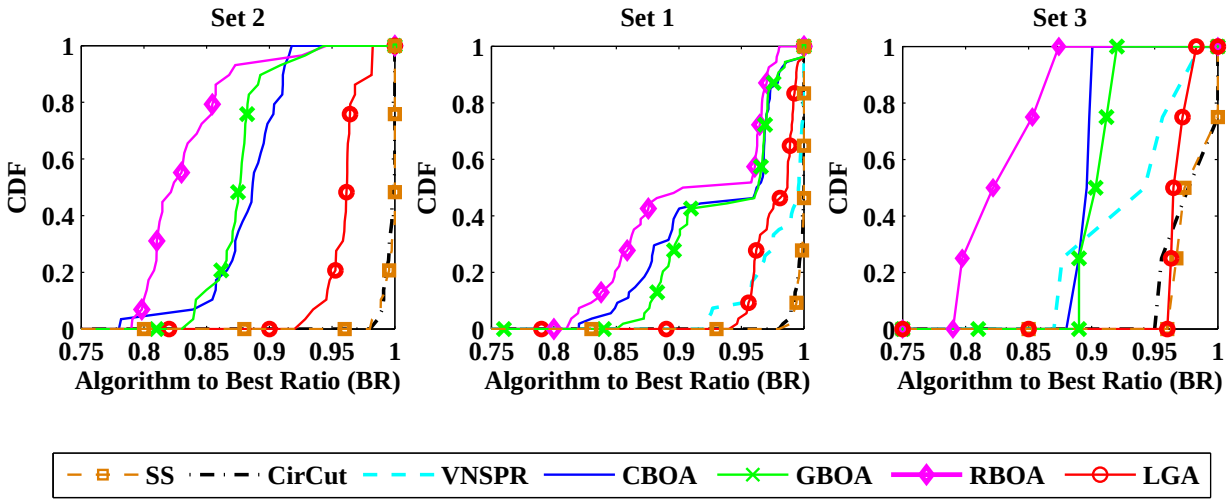


Figure 1: Max Cut Benchmark Algorithm Performance

We can see that out of all the methods we presented only LGA is competitive with the state of the art methods, and $BR_{LGA} \geq 92\%$ for all instances. In experiment set 3 LGA is better than $VNSPR$ and almost as good as $CirCut$. We see that $CBOA$ and $GBOA$ have almost the same performance, and that $RBOA$ is generally the worst. All methods gave LBR results better than $2/\pi$, despite the correction factors discussed above.

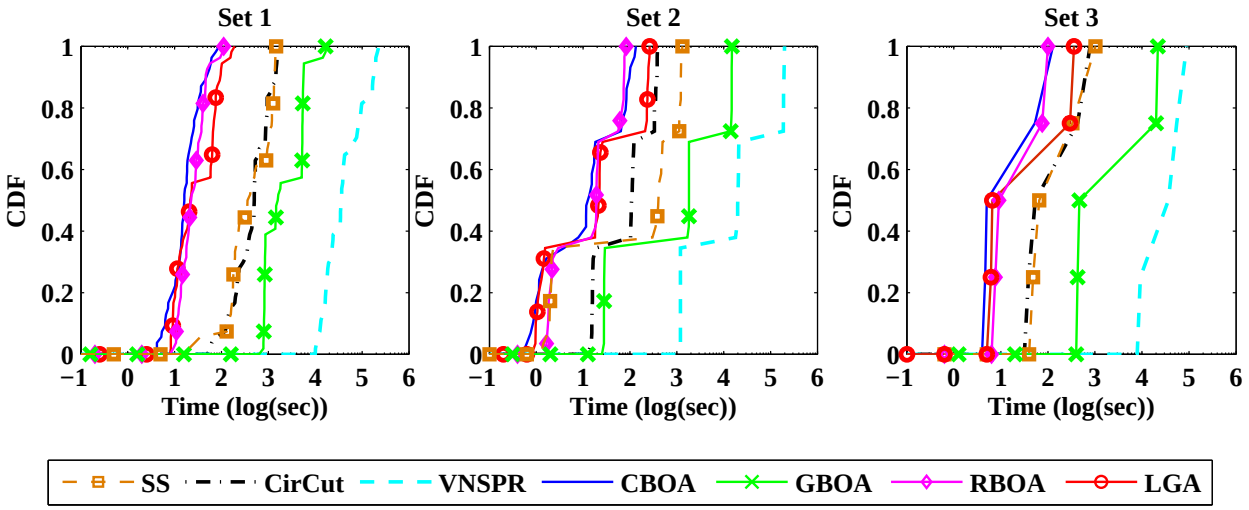


Figure 2: Max Cut Benchmark Algorithm Times

We also compare the algorithms' running times, to see if the theoretical convergence bounds are a good indication of the algorithms actual performance. Again the empirical CDF[1] of the running times is used as a basis for comparison, as shown in Figure 2. Here, since lower running times are better, the upper left CDF graphs will indicate better algorithms. We can see that the $VNSPR$ running time is 2-4 orders of magnitude larger than the other algorithms, hence value to running time it is the least effective. LGA running times, which include the solution of the SDP, are approximately less or equal the order of magnitude as those of SS and $CirCut$, and about the same as both $CBOA$ and $RBOA$.

Experiment 2 compares the performance of the various algorithms in Table 1 as a function of problem size (n), number of constraints (m), and objective function rank (r). For this purpose we looked at problem of size $n = 100, 200, 500, 1000$ for which we generated various PSD matrices of known rank $r \in [2, n]$. The constraints were constructed by a partition of the space to m equally sized blocks, $m \in [10, n]$ where each partition element was of size $\frac{n}{m}$ (for each n the number of

[1]The running times for algorithms SS, $CirCut$ and $VNSPR$ are taken from [30] and were run on a Dual Intel Xeon at 3.06 GHz with 3.2 GiB of RAM

blocks m was chosen so that values of $\frac{n}{m}$ were integers). For each 3-tuple (n, r, m) we conducted 5 simulations, each generated by sampling a random $n \times r$ matrix C for which $C_{ij} \in unif(-10, 10)$ and setting $\mathcal{A} = CC^T$.

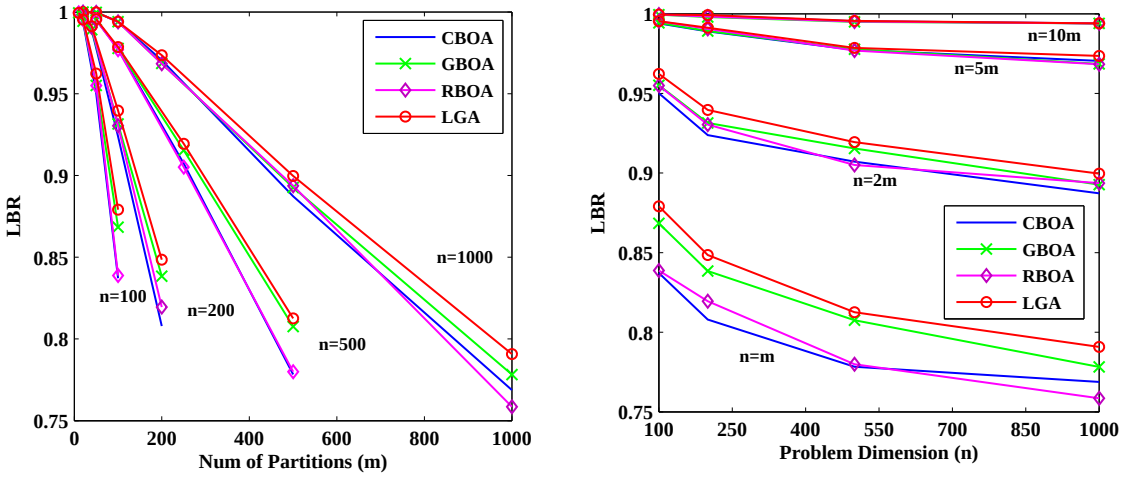
The SDP relaxation was attempted to be solved both by our modified $HRVW$ and by the CVX software [21], which solves generic SDP programs using the $SDPT3$ solver. For large problems ($n = 1000$) CVX took 30-40 minutes (we assume most of it was modeling time, since CVX does not allow to separate modeling and solver time) and resulted in an infeasible solution. For problems with $n \leq 500$ the CVX solution was 20-times slower than the $HRVW$ algorithm, and obtained the same objective function. For all realization, the solution matrix $\mathcal{X}$ always satisfied $rank(\mathcal{X}) \leq \min\{\sqrt{2m}, r\}$ and we were not able to reduce the rank further. We also checked if the accumulation point of the $CBOA$ satisfies the second order necessary conditions for local maximum and found that it did for all the realizations.

Figure 3 shows a the dependency between the LBR and both the problem dimension (n) and number of blocks (m). Figure 3(a) shows average LBR vs. m, for various problem dimensions. The average was taken over $r = 4, 10, 50, 100$ (total of 20 realizations for each value of n and m). We can clearly see the strong dependence and that indeed for all the algorithms LBR decreases as m increases. This behavior is replicated for all values of n, though, for lower values we see a steeper curve. This leads us to infer that the LBR is actually dependent on the ratio between n and m. This hypothesis is strengthened by the fact that for a constant m the LBR is increasing in n for all algorithms. Indeed, Figure 3(b) illustrates that LBR becomes a decreasing function of n when $\frac{n}{m}$ (block size) is held constant, which verifies our assumption. We did not find any consistent relation between the LBR and r in any of the methods, moreover, the impact of r on the LBR is negligible compared to that of m.

In order to better compare the algorithms to each other we use the *Average Ratio (AR)* measure. The AR is the ratio of the algorithm performance to the average performance,

$$AR_{alg} = |ALG| \frac{Val(Alg)}{\sum_{a \in ALG} Val(a)}$$

where ALG denotes the group of tested algorithms and $|ALG|$ is its cardinality. This measure



(a) (b)

Figure 3: (a) LBR vs. Number of Blocks, (b) LBR vs. Problem Dimension for Constant block size n/m

compares each algorithm to the average algorithm and indicates how much better/worse it is from the average. We also included a comparison with the proximal gradient (PGM) and greedy block proximal gradient methods (GBPGM), both with step size $t = L_f$. The PGM is not portrayed in the following graphs since its performance was significantly inferior to all other method, with AR always smaller than 1, and in most cases smaller than 0.95. A branch and bound algorithm based on [1], as described in section 1, was also tested on this scenarios. We ran the algorithm for 500 seconds. The results were much worse than any of the other algorithms (including PGM), with more than 85% of the realization with LBR and AR below 0.1. The larger n and n/m are the worse the performance of the methods, where n/m has a grater impact, for instance, for $n/m = 1$ (the MAX-CUT setting) all realizations have LBR smaller than 0.007.

Figure 4 depicts CDF of the AR measure, for all the experiments made for each algorithm for a certain block sizes $\frac{n}{m}$. We can clearly see that LGA is the superior algorithm and is better than average more than 90% of the time. $RBOA$ and $CBOA$ measure is almost identical and $GBOA$ performs slightly better.

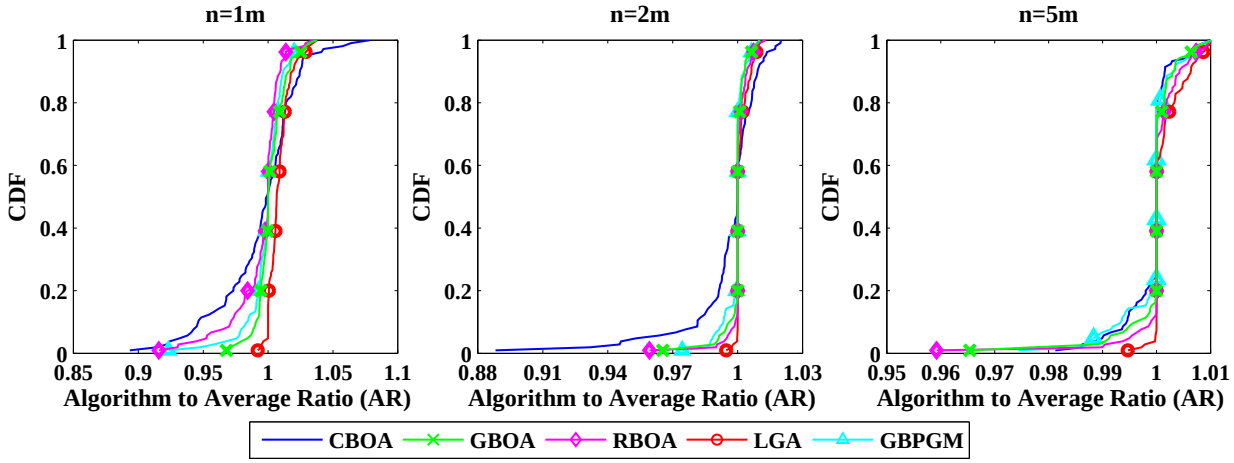


Figure 4: Block Experiments Performance for Constant n/m

Figure 5 shows the CDF of the algorithms running times. We can clearly see that $GBOA$ has the worst running time, while the other algorithms give similar running times, with a slight advantage to LGA and $CBOA$. Notice that even though $GBOA$ enables parallel computing techniques, we need approximately 100 times as many processors as the LGA for them to run the same amount of time. Furthermore, although the computational cost per iteration of $GBPGM$ is much smaller we can see that it takes more time than the other methods, this is due to the amount of iterations needed until the stopping criteria is met.

Over all LGA indeed gives better results, both in running times and AR measure especially for $n = m$. The LGA is the best method for over 70% of the instances (more than 90% when $n \geq 2m$), and is on average $2\% - 3\%$ better.

Conclusions We showed that LGA gives the best performance out of the algorithms discussed and presented in Table 1. It is also competitive with specifically tailored algorithms for the MAX-CUT problem. We also noticed that as block size decreases and number of blocks increases the tightness of the SDP and LGA solutions to the true optimal value becomes less certain. However, for block size of 2 and up, and problem size up to 1000 both methods give bounds which do not deviate more than 10% from the optimal value.

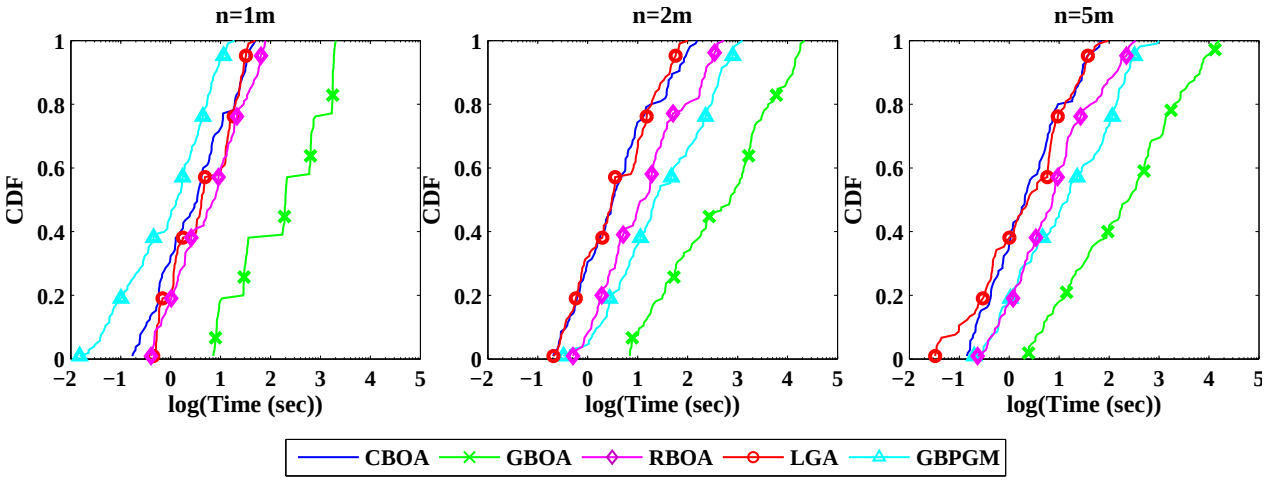


Figure 5: Block Experiments Times for Constant n/m

A Proof of theorem 5.2

Let $\mathcal{X}$ be a solution to the SDP relaxation of QCCCQP which satisfy all the constraints in equality. For the purpose of completeness, we will now repeat some of [4] proof in order to then show that in our case we can obtain a tighter bound on the rank of the optimal solution. Let matrix $\mathcal{X}$ be an optimal solution of the QCCCQP such that $\mathcal{B}_i \bullet \mathcal{X} = 1,\ i = 1, \ldots, m$ with objective function value is given by $\mathcal{A} \bullet \mathcal{X}$, where $\bullet$ is the matrix standard inner product. According to Barvinok [4], we can now look at the solution to the following problem:

$$\begin{aligned} &\max V^T \mathcal{A} V \bullet U \\ &s.t. \\ &\quad V^T \mathcal{B}_i V \bullet U \leq 1 \ \forall i = 1, \ldots, m \\ &\quad U \succeq 0 \end{aligned} \tag{49}$$

where V as some decomposition of $\mathcal{X} = VV^T$. It is clear that, for any solution U to this problem, we can construct a solution $\mathcal{X}(U) = VUV^T$ for the original problem. Specifically matrix I (the identity matrix) is an optimal solution for this problem (from optimality of $\mathcal{X}$). Therefore, any feasible solution, with the same objective function value, is optimal as well.

Without loss of generality, we may assume that $V^T\mathcal{A}V$ is a diagonal matrix, if it is not then there is a diagonalizing orthogonal matrix P, such that $PV^T\mathcal{A}VP^T$ is diagonal, and for any symmetric matrix $\mathcal{M}$ it hold that:

$$Tr(PV^T\mathcal{M}VP^T) = Tr(\mathcal{M}VP^TPV^T) = Tr(\mathcal{M}VV^T) = Tr(\mathcal{M}\mathcal{X}) = V^T\mathcal{M}V \bullet I,$$

and problem (49) can be written with $\tilde{V} = VP^T$ instead of V. Assuming $rank(\mathcal{A}) = r$ we have that $rank(V^T\mathcal{A}V) \leq r$ and we can assume w.l.o.g that only the first r diagonal elements are non-zero. We define the diagonal matrix I_r to have the first r diagonal elements equal 1 and all other zero (a $r \times r$ identity matrix padded with zeros). It is therefore obvious that $V^T\mathcal{A}V \bullet I_r = V^T\mathcal{A}V \bullet I$.

Notice that since $V^TB_iV \succeq 0$, and more specifically its diagonal is nonnegative, we have that $V^T\mathcal{B}_iV \bullet I_r \leq V^TB_iV \bullet I \leq 1,\ i = 1, \ldots, m$. Thus, I_r is both feasible and has the same objective function value as I, hence an optimal solution. Since $rank(I_r) = r$ we have that $rank(\mathcal{X}(I_r)) \leq r$, is an optimal solution to the original problem and the proof is concluded.

B Implementation of HRVW-alg to block case

We will now show how problem (PN) can be efficiently solved by the algorithm given in [23]. Looking at the constraints, we can define a general linear operator $B(\mathcal{X})$ such that we can write the constraints as $B(\mathcal{X}) \leq e$ where e is a column vector of ones, of length n. We also define $\chi(\mathcal{X}) = diag(\mathcal{X})$, where diag is the vector of diagonal entries, and $p_j = \mathcal{I}_j e$. Therefore, $Tr(\mathcal{I}_j X) = p_j^T d(\mathcal{X})$ and $B(X) = Pd(\mathcal{X})$ where P is a matrix with rows $\{p_j^T\}_{j=1}^m$. The conjugate operator $B^T(y)$ is defined such that $\left\langle B^T(y), X\right\rangle = \langle y, B(X)\rangle$ and in this case it is easy to see that $B^T(y) = \sum_{j=1}^{m}(y_j\mathcal{I}_j) = Diag(P^Ty)$, where $Diag$ transform a vector to the appropriate diagonal matrix.

Writing the Lagrangian for this problem we get that:

$$\begin{aligned} L(\mathcal{X}; y) &= tr(\tilde{\mathcal{A}}\mathcal{X}) - y^T(e - B(X)) \\ &= tr((\tilde{\mathcal{A}} - B^T(y))\mathcal{X}) + e^Ty \end{aligned} \tag{50}$$

Notice that the Lagrangian is bounded from above for any $\mathcal{X} \succeq 0$ iff $B^T(y) - \tilde{\mathcal{A}} \succeq 0$ in which case

for the optimal $\mathcal{X}$ it holds that: $tr((\tilde{\mathcal{A}} - B^T(y))\mathcal{X}) = 0$ and the dual SDP is given by:

$$\begin{aligned} &\min_y \{e^T y\} \\ &s.t. \quad B^T(y) - \tilde{\mathcal{A}} \succeq 0 \\ &\qquad y \geq 0 \end{aligned} \tag{51}$$

Since $\mathcal{A}$ is PSD we can actually drop the non-negativity constraint on y, or equivalently:

$$\begin{aligned} &\min_y \{e^T y\} \\ &s.t. \quad B^T(y) - \tilde{\mathcal{A}} = Z \\ &\qquad Z \succeq 0 \end{aligned} \tag{52}$$

We look at the associated barrier problem for (52) which is of the form:

$$\begin{aligned} &\min_y \{e^T y - \mu \log(det(Z))\} \\ &s.t. \quad B^T(y) - \tilde{\mathcal{A}} = Z \\ &\qquad Z \succeq 0 \end{aligned} \tag{53}$$

where $\mu > 0$ is a real positive parameter. The corresponding Lagrangian is given by:

$$L_\mu(y, Z; \mathcal{X}) = e^T y - \mu \log(det(Z)) + Tr(X(Z - B^T(y) + \tilde{\mathcal{A}})) \tag{54}$$

We obtain that:

$$\begin{aligned} \nabla_{\mathcal{X}} L_\mu &= Z - B^T(y) + \tilde{\mathcal{A}} = 0 \\ \nabla_y L_\mu &= e - B(\mathcal{X}) = 0 \\ \nabla_Z L_\mu &= -\mu Z^{-1} + \mathcal{X} = 0 \end{aligned} \tag{55}$$

Therefore, the primal-dual complementarity slackness is given by $Z\mathcal{X} = \mu I$ and $\mu = \frac{Tr(ZX)}{n}$ is the duality gap. In interior Primal-Dual methods we start from a matrix $\mathcal{X}^0 \succ 0$ which satisfies the strong inequality $B(\mathcal{X}) < b$ and dual variables Z^0, y^0 which satisfy the equations $Z^0 - B^T(y^0) + \tilde{\mathcal{A}} = 0$ and $Z^0 \succ 0$. In our problem such values are easy to find, simply choose any $\mathcal{X}^0$ such that $\mathcal{X}^0 = Diag(x)$ where $x = \sum_{j=1}^m \frac{1}{e^T p_j} p_j = \sum_{j=1}^m \frac{1}{n_j} p_j$. The resulting matrix $\mathcal{X}^0$ is therefore non-negative, diagonal, full rank, and consequently positive definite. Choosing $y > \|\mathcal{A}\| Pe$ will create $B^T(y^0) - \tilde{\mathcal{A}} = Z^0$ which diagonally dominant and so positive definite.

At each point in time the algorithm proceeds in the direction of the optimal solution to the barrier problem, to obtain a smaller duality gap for the original problem, and reduce μ. This direction, denoted by $(\Delta\mathcal{X}, \Delta Z, \Delta y)$, is a solution to the following linear equations, which are derived from (56) and the feasibility of the initial point $(\mathcal{X}, Z, y)$:

$$\begin{aligned}(Z+\Delta Z) - B^T(y+\Delta y) + \tilde{\mathcal{A}} = \Delta Z - B^T(\Delta y) = 0\\ e - B(\mathcal{X}+\Delta\mathcal{X}) = B(\Delta\mathcal{X}) = 0\\ -\mu I + (Z+\Delta Z)(\mathcal{X}+\Delta\mathcal{X}) \approx (-\mu I + Z\mathcal{X}) + \Delta Z\mathcal{X} + Z\Delta\mathcal{X} = 0\end{aligned} \tag{56}$$

where the last equation the term $(\Delta Z\Delta\mathcal{X})$ is neglected. In [23] the authors present the solution which, in problem (PN), can be simply written as:

$$\begin{aligned}\Delta Z &= B^T(\Delta y)\\ \Delta\mathcal{X} &= \mu Z^{-1} - \mathcal{X} + (Z^{-1}B^T(\Delta y)\mathcal{X})\\ \Delta y &= (P(Z^{-1}\odot\mathcal{X})P^T)^{-1}P(\mu diag(Z^{-1}) - diag(\mathcal{X}))\end{aligned} \tag{57}$$

where $\odot$ denotes a component-wise, or Hadamard, product. Notice also that matrix $\mathcal{P} = P(Z^{-1}\odot\mathcal{X})P^T$ is an $m\times m$ matrix, with components $\mathcal{P}_{i,j} = tr(\mathcal{I}_i(Z^{-1}\odot\mathcal{X})\mathcal{I}_j) = \mathcal{I}_i X\mathcal{I}_j \bullet \mathcal{I}_i Z^{-1}\mathcal{I}_j$ where $\bullet$ is the inner product of matrices and $\mathcal{I}_i(\cdot)\mathcal{I}_j$ denotes the i,j block of the matrix. Calculating this matrix takes $\sum_{i=1}^m\sum_{j=1}^m n_i n_j = n^2$ operations.

To show that this matrix is indeed invertible, we use the fact that Z and $\mathcal{X}$ are both PD, and symmetric, and so $(Z^{-1}\odot\mathcal{X})$ is both symmetric and PSD since

$$\begin{aligned}u^T(Z^{-1}\odot\mathcal{X})u &= \sum_{i,j} u_i Z^{-1}_{i,j}\mathcal{X}_{i,j}u_j = \sum_{i,j} u_i Tr(Diag(e_i)Z^{-1}Diag(e_j)\mathcal{X})u_j\\ &= \sum_{i,j} Tr(\mathcal{X}^{\frac{1}{2}}Diag(e_i)Z^{-1}Diag(e_j)\mathcal{X}^{\frac{1}{2}})u_i u_j\\ &= Tr(\sum_{i,j} u_i\mathcal{X}^{\frac{1}{2}}Diag(e_i)Z^{-1}Diag(e_j)\mathcal{X}^{\frac{1}{2}}u_j)\\ &= Tr((\sum_i u_i\mathcal{X}^{\frac{1}{2}}Diag(e_i)Z^{-\frac{1}{2}})(\sum_j u_j\mathcal{X}^{\frac{1}{2}}Diag(e_j)Z^{-\frac{1}{2}}))^T \geq 0\end{aligned} \tag{58}$$

Notice that denoting $M = \sum_i u_i\mathcal{X}^{\frac{1}{2}}Diag(e_i)Z^{-\frac{1}{2}} = \mathcal{X}^{\frac{1}{2}}Diag(u)Z^{-\frac{1}{2}}$ and since from $u\neq 0$ and both $\mathcal{X}$ and Z are full rank matrices (PD matrices) we obtain that $M\neq 0$. Therefore, $Tr(MM^T)\neq 0$

for any $u \neq 0$ and consequently the matrix $Z^{-1} \odot \mathcal{X}$ is PD. Since for any $u \in \mathbb{R}^m$ such that $u \neq 0$ we have $v \in \mathbb{R}^n$ such that $v = P^T u \neq 0$ we can conclude $P(Z^{-1} \odot \mathcal{X})P^T$ is PD.

While computing equation (57) We need to maintain the symmetry and positive definiteness of the resulting matrices. Since $\Delta\mathcal{X}$ might not be symmetric, corrected $\Delta\tilde{\mathcal{X}} = \frac{\Delta\mathcal{X}+\Delta\mathcal{X}^T}{2}$ is used. As for the positive definiteness, we find two positive scalars $0 < \alpha_D < 1$ and $0 < \alpha_P < 1$, such that:

$$\begin{aligned} Z^{new} &= Z + \alpha_D \Delta Z \succ 0 \\ y^{new} &= y + \alpha_D \Delta y \\ \mathcal{X}^{new} &= \mathcal{X} + \alpha_P \Delta\mathcal{X} \succ 0 \end{aligned} \tag{59}$$

Some backtracking can be used to determine α_P and α_D, while maintaining feasibility and decreasing the duality gap.

As we stated before, the most computationally expensive operation we do is the matrix inversion, which takes $O(n^3)$ operations for matrix Z, $O(m^3)$ operations for matrix $\mathcal{P}$. The creation of matrix $\mathcal{P}$ costs only $O(n^2)$ operations regardless of m. When $m = n$ things simplify even further since $P = I$. Moreover, according to [23] the bound on the number of iteration needed for a given dual gap ϵ is $O(\sqrt{n}\log\left(\frac{Tr(\mathcal{X}^0 Z^0)}{\epsilon}\right))$, independent of m.

C Proof of lemma 7.2

(i) We know that if point x is block-wise optimal, it satisfies the block-wise first order necessary conditions for optimality (see [7]), and since the block-wise problems are also regular, the multipliers are unique. i.e. $\exists \mu_i \geq 0$:

$$\begin{aligned} L_i(x_i; \mu_i | x_{-i}) &= x_i^T \tilde{A}_i^T \tilde{A}_i x_i + 2x_i^T \tilde{A}_i^T \tilde{A}_{-i} x_{-i} + x_{-i}^T \tilde{A}_{-i}^T \tilde{A}_{-i} x_{-i} - \mu_i (x_i^T x_i - 1) \\ \nabla_{x_i} L_i(x_i; \mu_i | x_{-i}) &= 2(\tilde{A}_i \tilde{E} x - \mu_i x_i) = 0 \\ \mu_i(\|x_i\|^2 - 1) &= 0 \\ \|x_i\|^2 &\leq 1 \end{aligned} \tag{60}$$

We do not need the complementarity constraints since $x_i^T x_i = 1$ and μ_i is uniquely defined by $\mu_i = x_i^T A_i^T \tilde{E} x$.

Looking at the first order necessary optimality conditions for problem (PN) (omitting complementarity and feasibility constraints):

$$\begin{aligned} L(x;\mu) &= x^T\tilde{E}^T\tilde{E}x - \sum_{i=1}^{m}\mu_i(x^T\mathcal{I}_i x - 1) \\ \nabla_x L(x;\mu) &= 2(\tilde{E}^T\tilde{E}x - \sum_{i=1}^{m}\mu_i\mathcal{I}_i x) = 0 \end{aligned} \tag{61}$$

Notice the second constraint is equivalent to $\tilde{A}_i^T\tilde{E}x - \mu_i x_i = 0\ \forall i = 1,\dots,m$ and so the unique multipliers μ_i which satisfy (60), must also satisfy (61).

(ii) If x is globally optimal then $F(x) \geq F(y)$ for any feasible $y \in C$. More specifically: $F_i(x_i|x_{-i}) \geq F(y_i|x_{-i})$ for all $y_i \in C_i$ and so x is block-wise globally optimal.

(iii) From the block optimality of x_i we obtain:

$$F(x) = F_i(x_i|x_{-i}) \geq F_i(z_i|x_{-i}) \quad \forall z_i : z_i^T z_i \leq 1 \tag{62}$$

and specifically we have :

$$x_i^T\tilde{A}_i^T\tilde{A}_i x_i + 2x_i^T\tilde{A}_i^T\tilde{A}_{-i}x_{-i} \geq \|A_i\|^2 > 0 \tag{63}$$

From the first order conditions combined with the constraint being satisfied in equality we get that $\mu_i \geq \|A_i\|^2 > 0$.

The second order necessary optimality conditions (see [7]) for problem (PN) state that for any d such that

$$d \in D = \{d_i^T x_i \leq 0\ \forall i = 1,\dots,m,\ d^T\tilde{E}^T\tilde{E}x \geq 0\}$$

there exist multipliers $\mu \in \mathbb{R}^m$ such that

$$\begin{aligned} \mu_i x_i = \tilde{A}_i^T\tilde{E}x, \quad \mu_i d_i^T x_i = 0\ \forall i = 1,\dots,m \\ d^T\tilde{\mathcal{A}}x = 0 \\ d^T(\tilde{E}^T\tilde{E} - \sum_{i=1}^{m}\mu_i\mathcal{I}_i)d = d^T(\tilde{\mathcal{A}} - \sum_{i=1}^{m}\mu_i\mathcal{I}_i)d \leq 0. \end{aligned} \tag{64}$$

Since $\mu_i > 0$ and unique, the necessary conditions are equivalent to solving the following problem: $\max_d\{d^T(\tilde{\mathcal{A}} - \sum_{i=1}^{m} \mu_i \mathcal{I}_i)d : d_i^T x_i = 0 \,\forall i = 1, \ldots, m,\ \|d\| \leq 1\}$. It is of course obvious that if $P = \tilde{\mathcal{A}} - \sum_{i=1}^{m} \mu_i \mathcal{I}_i$ is a NSD matrix the necessary conditions hold by default, and if P is a ND matrix the sufficient conditions hold, so let us assume that this is not the case. Since we know $\|x_i\| = 1$ there exists at least one component j_i in each x_i such that $x_i(j_i) \neq 0$ and so we can rewrite $d_i^T x_i = 0$ as $d_i(j_i) = -\sum_{k \neq j_i} \frac{x_i(k) d_i(k)}{x_i(j_i)}$ (notice that for $n_i = 1$ we have that $d_i = 0$ for all i and we are at a local minimum). Without loss of generality, we will assume $j_i = n_i \,\forall i$. We will define

$$\mathcal{T}_i = \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \\ -\frac{x_i(1)}{x_i(n_i)} & \cdots & -\frac{x_i(n_i-1)}{x_i(n_i)} \end{bmatrix}, \quad \mathcal{T} = \begin{bmatrix} \mathcal{T}_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \mathcal{T}_m \end{bmatrix} \in \mathbb{R}^{n \times (n-m)}$$

and $d = \mathcal{T} c$ or equivalently $d_i = \mathcal{T}_i c_i,\ i = 1, \ldots, m$. Clearly, $\mathcal{T}_i \in \mathbb{R}^{n_i \times (n_i - 1)}$ and, consequently, $\mathcal{T}$ are full column rank matrices. Therefore, we can rewrite the problem as follows:

$$\max_c\{c^T \mathcal{T}^T (\tilde{\mathcal{A}} - \sum_{i=1}^{m} \mu_i I_i') \mathcal{T} c : \|\mathcal{T} c\| \leq 1\}$$

Since matrix $\tilde{\mathcal{T}} = \mathcal{T}^T \mathcal{T}$ is positive definite we can use its Cholesky decomposition G to transform the problem to the equivalent problem of finding the eigenvector $\tilde{c}$ of matrix $\tilde{P} = G^{-T} \mathcal{T}^T (\tilde{\mathcal{A}} - \sum_{i=1}^{m} \mu_i \mathcal{I}_i) \mathcal{T} G^{-1}$ corresponding with its maximal eigenvalue. If the maximum eigenvalue is positive than the necessary conditions do not hold. This means that given $d = C G^{-1} \tilde{c}$ there exists a vector z such that the following linear inequalities are satisfied:

$$\begin{aligned} x_i^T z_i + d_i^T d_i + \alpha \leq 0,\ \forall i = 1, \ldots, m \\ z^T \tilde{\mathcal{A}} x + d \tilde{\mathcal{A}} d - \alpha \geq 0 \end{aligned} \tag{65}$$

for some $\alpha > 0$ (we can simply solve with objective function $\max \alpha$). We choose a new point $\tilde{x} = x + td + \frac{t^2}{2} z$ for some $t > 0$ which satisfies:

$$\begin{aligned} (x_i + t d_i + \frac{t^2}{2} z_i)^T (x_i + t d_i + \frac{t^2}{2} z_i) \leq 1,\ \forall i = 1, \ldots, m \\ (x + td + \frac{t^2}{2} z)^T \tilde{\mathcal{A}} (x + td + \frac{t^2}{2} z) \geq x^T \tilde{\mathcal{A}} x \end{aligned} \tag{66}$$

Since z satisfy the constraints 65 it follows that we can always find $t > 0$ which satisfy 66. This new point is not necessarily stationary and so we can use some procedure to obtain a better stationary point, which is also an extreme point.

References

[1] F. Al-Khayyal, C. Larsen, and T. Voorhis, *A relaxation method for nonconvex quadratically constrained quadratic programs*, J. Global Optim., 6 (1995), pp. 215–230.

[2] H. Attouch, J. Bolte, P. Redont, and A. Soubeyran, *Proximal alternating minimization and projection methods for nonconvex problems: An approach based on the Kurdyka-Łojasiewicz inequality*, Math. Oper. Res., 35 (2010), pp. 438–457.

[3] F. Barahona, M. Grötschel, M. Jünger, and G. Reinelt, *An application of combinatorial optimization to statistical physics and circuit layout design*, Oper. Res., 36 (1988), pp. 493–513.

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

[5] A. Beck and M. Teboulle, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems*, SIAM J. Imaging. Sci., 2 (2009), pp. 183–202.

[6] A. Beck and L. Tetruashvili, *On the convergence of block coordinate descent type methods*, SIAM J. Optim., 23 (2013), pp. 2037–2060.

[7] A. Ben-Tal, *Second-order and related extremality conditions in nonlinear programming*, J. Optim. Theory Appl., 31 (1980), pp. 143–165.

[8] A. Ben-tal, A. Nemirovski, and C. Roos, *Extended matrix cube theorems with applications to μ-theory in control*, Math. Oper. Res., 28 (2003), pp. 497–523.

[9] A. Ben-Tal and M. Teboulle, *Hidden convexity in some nonconvex quadratically constrained quadratic programming*, Math. Program., 72 (1996), pp. 51–63.

[10] H. P. Benson and R. Horst, *A branch and bound-outer approximation algorithm for concave minimization over a convex set*, Comput. Math. Appl., 21 (1991), pp. 67 – 76.

[11] D. P. Bertsekas and P. Tseng, *Partial proximal minimization algorithms for convex programming*, SIAM J. Optim., 4 (1994), pp. 551–572.

[12] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation : numerical methods*, Prentice-Hall, Upper Saddle River, NJ, USA, 1989.

[13] J. Bolte, S. Sabach, and M. Teboulle, *Proximal alternating linearized minimization for nonconvex and nonsmooth problems*, Math. Programming, (2013), pp. 1–36.

[14] A. De Maio, S. De Nicola, Y. Huang, S. Zhang, and A. Farina, *Code design to optimize radar detection performance under accuracy and similarity constraints*, IEEE Trans. Signal Process., 56 (2008), pp. 5618–5629.

[15] A. De Maio and A. Farina, *Code selection for radar performance optimization*, in Waveform Diversity and Design Conference, 2007. International, Pisa, Italy, 2007, pp. 219–223.

[16] R. Elsässer and T. Tscheuschner, *Settling the complexity of local max-cut (almost) completely*, in Automata, Languages and Programming, L. Aceto, M. Henzinger, and J. Sgall, eds., vol. 6755 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 171–182.

[17] C. Floudas and C. Gounaris, *A review of recent advances in global optimization*, J. Global Optim., 45 (2009), pp. 3–38.

[18] M. X. Goemans and D. Williamson, *Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming*, J. ACM, 42 (1995), pp. 1115–1145.

[19] M. X. Goemans and D. P. Williamson, *Approximation algorithms for max-3-cut and other problems via complex semidefinite programming*, J. Comput. System Sci., 68 (2004), pp. 442–470.

[20] G. H. Golub and C. F. Van Loan, *Matrix computations*, Johns Hopkins University Press, Baltimore, MD, USA, 1996.

[21] M. Grant and S. Boyd, *CVX: Matlab software for disciplined convex programming, version 1.21*. ../../cvx, Apr. 2011.

[22] L. Grippo and M. Sciandrone, *On the convergence of the block nonlinear gaussseidel method under convex constraints*, Oper. Res. Lett., 26 (2000), pp. 127–136.

[23] C. Helmberg, F. Rendl, R. J. Vanderbei, and H. Wolkowicz, *An interior-point method for semidefinite programming*, SIAM J. Optim., 6 (1996), pp. 342–361.

[24] R. A. Horn and C. R. Johnson, *Matrix analysis*, Cambridge University Press, New York, NY, USA, 1985.

[25] R. Horst, *An algorithm for nonconvex programming problems*, Math. Programming, 10 (1976), pp. 312–321.

[26] E. S. Levitin and B. T. Polyak, *Constrained minimization methods*, U.S.S.R. Comput. Math. Math. Phys., 6 (1966), pp. 1–50.

[27] Z.-Q. Luo, W.-K. Ma, A.-C. So, Y. Ye, and S. Zhang, *Semidefinite relaxation of quadratic optimization problems*, IEEE Signal Process. Mag., 27 (2010), pp. 20–34.

[28] Z.-Q. Luo and P. Tseng, *Error bounds and convergence analysis of feasible descent methods: a general approach*, Ann. Oper. Res., 46-47 (1993), pp. 157–178.

[29] A. S. Man-cho, J. Zhang, and Y. Ye, *On approximating complex quadratic optimization problems via semidefinite programming relaxations*, Math. Programming, 110 (2007), pp. 93–110.

[30] R. Martí, A. Duarte, and M. Laguna, *Maxcut problem*. http://www.optsicom.es/maxcut/, 2009.

[31] B. Martinet, *Brve communication. rgularisation d'inquations variationnelles par approximations successives*, ESAIM, Math. Model. Numer. Anal., 4 (1970), pp. 154–158.

[32] J. Moré and D. Sorensen, *Computing a trust region step*, SIAM J. Sci. Stat. Comp., 4 (1983), pp. 553–572.

[33] A. Nemirovski, C. Roos, and T. Terlaky, *On maximization of quadratic form over intersection of ellipsoids with common center*, Math. Programming, 86 (1999), pp. 463–473.

[34] Y. Nesterov, *Global quadratic optimization via conic relaxation*, CORE Discussion Papers 1998060, Universit catholique de Louvain, Center for Operations Research and Econometrics (CORE), Louvain-la-Neuve, Belgium, 1998.

[35] Y. Nesterov, *Semidefinite relaxation and nonconvex quadratic optimization*, Optim. Methods Softw., 9 (1998), pp. 141–160.

[36] Y. Nesterov, *Gradient methods for minimizing composite functions*, Math. Programming, 140 (2013), pp. 125–161.

[37] J. Nocedal and S. J. Wright, *Numerical optimization*, Springer Series in Operations Research, Springer, New York, NY, USA, 1999.

[38] G. Pataki, *On the rank of extreme matrices in semidefinite programs and the multiplicity of optimal eigenvalues*, Math. Oper. Res., 23 (1998), pp. 339–358.

[39] S. Poljak, *Integer linear programs and local search for max-cut*, SIAM J. Comput., 24 (1995), pp. 822–839.

[40] M. J. D. Powell, *On search directions for minimization algorithms*, Math. Programming, 4 (1973), pp. 193–201.

[41] R. T. Rockafellar, *Monotone operators and the proximal point algorithm*, SIAM J. Control Optim., 14 (1976), p. 877.

[42] S. Sahni and T. Gonzalez, *P-complete approximation problems*, J. ACM, 23 (1976), pp. 555–565.

[43] A. Shapiro, *Rank-reducibility of a symmetric matrix and sampling theory of minimum trace factor analysis*, Psychometrika, 47 (1982), pp. 187–199.

[44] D. Sorensen, *Newtons method with a model trust region modification*, SIAM J. Numer. Anal., 19 (1982), pp. 409–426.

[45] P. Tseng, *Convergence of a block coordinate descent method for nondifferentiable minimization 1*, J. Optim. Theory Appl., 109 (2001), pp. 475–494.

[46] Y. Ye, *Approximating quadratic programming with bound and quadratic constraints*, Math. Programming, 84 (1999), pp. 219–226.

[47] S. Zhang, *Quadratic maximization and semidefinite relaxation*, Math. Programming, 87 (2000), pp. 453–465.

[48] S. Zhang and Y. Huang, *Complex quadratic optimization and semidefinite programming*, SIAM J. Optim., 16 (2006), pp. 871–890.

[49] X. Zheng, X. Sun, and D. Li, *Nonconvex quadratically constrained quadratic programming: best D.C. decompositions and their SDP representations*, J. Global Optim., 50 (2011), pp. 695–712.