A Fast Binary Quadratic Programming Solver based on Stochastic Neighborhood Search

Benson Shu Yan Lam and Alan Wee-Chung Liew, Senior Member, IEEE

Abstract—Many image processing and pattern recognition problems can be formulated as binary quadratic programming (BQP) problems. However, solving a large BQP problem with a good quality solution and low computational time is still a challenging unsolved problem. Current methodologies either adopt an independent random search in a semi-definite space or perform search in a relaxed biconvex space. However, the independent search has great computation cost as many different trials are needed to get a good solution. The biconvex search only searches the solution in a local convex ball, which can be a local optimal solution. In this paper, we propose a BQP solver that alternatingly applies a deterministic search and a stochastic neighborhood search. The deterministic search iteratively improves the solution quality until it satisfies the KKT optimality conditions. The stochastic search performs bootstrapping sampling to the objective function constructed from the potential solution to find a stochastic neighborhood vector. These two steps are repeated until the obtained solution is better than many of its stochastic neighborhood vectors. We compare the proposed solver with several state-of-the-art methods for a range of image processing and pattern recognition problems. Experimental results showed that the proposed solver not only outperformed them in solution quality but also with the lowest computational complexity.

Index Terms—Binary Quadratic Programming, Stochastic Optimization, Binary Restoration, Graph Bisection, Optimization

1 Introduction

Many image processing, pattern recognition, and computer vision problems can be formulated as binary quadratic programming (BQP) problems, including image segmentation, image restoration, graph partitioning, data clustering, and data classification. An example is the image segmentation problem, in which an image is divided into two parts [1]. One is the object of interest, and the other is not. The binary variables are used to represent the object of interest and non-object of interest. This problem then becomes a binary programming problem. Another example is the dimensionality reduction problems. Recently, $l_1$ – PCA and $l_1$ – LDA models [2-4] have attracted great attention in various fields owing to their robustness towards noise and outliers. Their formulation usually involves $l_1$ norm, in which its dual form is a binary variable. The dual formulation is a BQP problem [2,5]. Here, the binary variable indicates if a projected sample is above the hyper-plane or below the hyper-plane. Other examples include the character recognition and fingerprint recognition problems, where the inputs are binary images. The input images of these problems are usually noisy and regularization is applied to restore the binary image. This can be formulated as a binary quadratic programming problem. The binary programming problem can also be extended to solve the Markov random field inference problems [6,7] in which the labels serve as the binary variables. Besides, the binary variable can also be used as an indicator to select the most effective features [8].

The very first BQP problem goes back to the 1960s, when the topics of pseudo-Boolean functions and the optimization technique were introduced [9]. However, the nature of binary solutions generally makes the problem NP-hard. Many different methods have been proposed, including exact solvers, heuristic solvers, and relaxed solvers [10]. For the exact solvers, the methods attempt to find the global optimal solution to the problem. The common approach under this category involves a tree search of a general branch-and-bound technique [11,12]. Recently, Mauri and Lorena proposed a Lagrangian decomposition method to handle the problem that incorporates graph partitioning to form subproblems that are solved by CPLEX [13]. However, these solvers can only handle a problem with up to 1000 variables. To handle larger-scale problems, heuristic solvers are introduced. In this category, tabu search [14,15], genetic algorithm [16], and simulated annealing [17] are widely used to find high-quality solutions. One example is the multilevel algorithm introduced by Wang et. al. [18]. This algorithm is composed of different phases of search, including a backbone-based coarsening phase, an asymmetric uncoarsening phase, and a memetic refinement phase. The phases incorporate tabu search to obtain improved solutions. Studies have shown that this type of method can handle problems with up to 15,000 variables.

In addition to exact and heuristic solvers, relaxed solvers are widely used, especially in the areas of image processing and pattern recognition. They can be broadly classified as spectral method, semidefinite relaxation (SDR) and the exact penalty method (EPM). The idea of spectral relaxation is to relax the binary constraint (i.e., $s = \{-1,1\}^n$) into a spherical constraint (i.e., $\|s\|^2 = 1$) [19-24]. Because the spectral method is simple, the solution can be obtained
very quickly. This technique has therefore been widely applied to solve various problems, including motion segmentation, clustering, and digital matting [1,19,22,24]. However, the loose bound nature of the spectral method can lead to poor quality solution in many cases [20,21,23]. Olson et al. used a sub-gradient approach with a trust region technique to improve the bounds of the spectral method [25,26]. This enhanced method can produce better results; however, it is sensitive to the user-defined parameters [25,26].

The SDR transforms the binary constraints into a semidefinite constraint [27,28]. This turns the problem into a semi-definite programming problem, which is convex and has a global optimal solution. The random hyperplane rounding technique [29] is then applied to the obtained semi-definite matrix to obtain an optimal binary vector for the original BQP problem. The strength of this approach is that its bound approximation is tighter than that of the spectral method and it thus gives a very-high quality solution [30]. SeDuMi and SDPT3 are two examples of SDR method [31,32]. SeDuMi performs optimization over symmetric cones, whereas SDPT3 adopts an infeasible path-following algorithm to solve the problem. However, to satisfy the semidefinite constraint, these two solvers make use of the interior point methods, which leads to poor scalability for large-scale problems [33]. The worst-case complexity of solving these problems is about \(O(n^{6.5})\). More recently, Wang et al. proposed a new technique called SDCut to handle the relaxation problems [34,35]. In addition to the semidefinite constraint, they introduced a special spherical constraint that can simplify the computation using eigenvalue decomposition. The computational complexity of the key steps of SDCut is \(O(kn^3)\), where \(k\) is the number of times the eigenvalue decomposition is applied. However, for real-world image processing and computer vision problems, \(O(kn^3)\) is still a great computational burden. In addition to high computation, SDR also has a high storage requirement. Since the relaxation turns the binary vector into a matrix, the number of unknowns increases significantly. For example, if the original problem has \(10^4\) variables, the number of variables for the relaxation will be \(10^8\). After the SDR step, the random hyperplane rounding technique is applied to obtain the binary solution. This involves an independent search over the semi-definite space by generating a set of random vectors based on a multivariate normal distribution with zero mean and a covariance matrix given by the obtained semi-definite matrix. The signed random vector that gives the best objective function value is the optimal solution. As the random vectors are independent of each other, many trials are needed to get an optimal solution. Moreover, even if a random vector is close to the global optimal solution, the next binary vector can be far away from it.

The EPM transforms the binary constraints to a set of inequality constraints [36-39]. Yuan et al. [41] substituted the binary constraints by an \(l_2\) box constraint. This turns the BQP problem into a biconvex problem. The optimal solution can then be obtained by applying standard optimization techniques such as gradient descent or even Newton method. However, owing to the biconvex nature of the problem, the obtained solution is only optimal in a local convex ball. If the initial guess is far away from the good subspace, this method may converge to a lower quality solution. A trial of different initial guess is needed to get to a good solution.

In this paper, we present a novel BQP solver that search for the locally optimal solution over stochastic neighborhoods. Our method performs a bootstrapping procedure to a potential solution to obtain a stochastic neighborhood vector. Then, a deterministic search is performed on the stochastic neighborhood vector to obtain a new solution. If a better solution is obtained, this will be treated a new potential solution. This procedure is repeated until the solution is better than many of its stochastic neighborhood vectors. In other words, the obtained solution is optimal with respect to the set of stochastic neighborhood vectors, which represents a broad range of feasible solutions. In contrast to SDR, the proposed solver performs a dependent search. That is, the new solution is dependent on the previous solutions. If the potential solution is close to the global optimal solution, the proposed solver can quickly perform a local search for the global optimal solution in its stochastic neighborhood. Unlike EPM, the stochastic neighborhood vectors obtained by the proposed random technique are not restricted to a local convex ball. Our method significantly enlarges the search region and ensures that the solution is optimal in a broader feasible region. We apply the proposed solver to several image processing, pattern recognition, and computer vision applications that can be formulated as unconstrained, equality constrained, or inequality constrained problems. Experimental results show that the proposed solver outperforms many state-of-the-art solvers. The main contributions of this paper are as follows.

**Deterministic Search**: We propose the use of the projected gradient ascent method to find a local optimal solution of the BQP problem. We mathematically prove that the method converges and also the obtained solution satisfies all the KKT optimality conditions. This allows the solution to be of high quality.

**Stochastic Neighborhood**: We propose a novel bootstrapping approach to find the stochastic neighborhood vectors of the potential optimal solution. When combined with deterministic search, the obtained solution is optimal in a large search region. In optimization theory, a solution that is optimal in a larger search region means there is a higher chance to get a better solution.

**Efficient Implementation**: The proposed method solves a BQP problem with low computational cost. In fact, it only involves matrix-vector multiplication in most steps. Thus, its computational complexity is only \(O(n^3)\). This is of the same order as the size of the input matrix.

The rest of the paper is organized as follows. In Section 2, we give a brief review of the BQP problems and the widely used relaxation techniques. In Section 3, we introduce the proposed solver, which consists of two main
parts: deterministic search and stochastic neighborhood vector generation. Experiments to verify the robustness of the proposed solver are shown in Section 4. Finally, our conclusions and directions for future work are given in Section 5.

<table>
<thead>
<tr>
<th>Abbreviations</th>
<th>General BQP</th>
<th>BQP</th>
<th>U.BQP</th>
<th>C.BQP</th>
<th>EPM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spectral Relaxation</td>
<td>Spectral Relaxation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Notation** A matrix is represented by a bold capital letter \( \mathbf{X} \), and its \((i,j)\)th entry is represented by \( x_{ij} \). A column vector is represented by a bold lower-case letter \( \mathbf{x} \), and its ith entry is represented by \( x_i \). A lower-case letter \( x \) is a scalar. The symbol \( \mathbf{Q} \) represents the \( n \times n \) symmetric matrix for all BQP problems. \( \mathbf{A}_{ineq}, \mathbf{A}_{eq}, \mathbf{b}_{ineq}, \mathbf{b}_{eq} \) are coefficients for the linear inequality and linear equality constraints and the corresponding right-hand side values for the linear inequality and linear equality constraints respectively. \( \mathbf{L}_{ineq} \) and \( \mathbf{L}_{eq} \) are the vectors of Lagrange multipliers for the inequality and equality constraints. \( \mathbf{s} = [s_1, s_2, ..., s_n]^T \) indicates a vector with elements of either +1 or -1. \( \mathbf{l}_{ineq}, \mathbf{l}_{eq} \) and \( \mathbf{s} \) are \( \mathbf{L}_{ineq}, \mathbf{L}_{eq} \) and \( \mathbf{s} \) at the t-iteration. sign(\( \mathbf{x} \)) is the vector with the signs of the elements of \( \mathbf{x} \). 0 and 1 indicate vectors with all zeros and ones, respectively. \( \Omega \) represents the feasible solutions. Lastly, the operators \( \cdot \) and \( \otimes \) represent elementwise multiplication and division between two vectors respectively.

2 REVIEW OF BQP PROBLEMS AND WIDELY USED RELAXATIONS

In this paper, we focus on the general BQP problem with linear equality and inequality constraints. It is written as

\[
\begin{align*}
\text{max}_{s \in \{\pm 1\}^n} \langle \mathbf{s} \rangle &= \mathbf{s}^T \mathbf{Qs} \quad \text{s.t. } \mathbf{s} \in \Omega, \\
\text{with } \Omega &= \{ \mathbf{s} | \mathbf{A}_{ineq} \mathbf{s} \leq \mathbf{b}_{ineq}, \mathbf{A}_{eq} \mathbf{s} = \mathbf{b}_{eq} \}.
\end{align*}
\]

A solution is said to be a local optimal solution of this problem if it satisfies the KKT optimality conditions below [33]:

\[
\begin{align*}
\text{(KKT-stationarity)} & \quad \mathbf{Ds} = \mathbf{Qs} + \mathbf{L}_{ineq}^T \mathbf{A}_{ineq} + \mathbf{L}_{eq}^T \mathbf{A}_{eq} s \in \{\pm 1\}^n, \\
\text{(KKT-primal feasibility)} & \quad \mathbf{A}_{ineq} \mathbf{s} \leq \mathbf{d}_{ineq}, \mathbf{A}_{eq} \mathbf{s} = \mathbf{b}_{eq}, \\
\text{(KKT-complementary slackness)} & \quad \mathbf{L}_{ineq} \mathbf{s} = \mathbf{b}_{ineq}, \\
\text{(KKT-dual feasibility)} & \quad \mathbf{L}_{ineq} \mathbf{s} \leq 0.
\end{align*}
\]

where the matrix \( \mathbf{D} \) is a diagonal matrix with the Lagrange multipliers in the diagonal entries for the binary constraint. \( \mathbf{L}_{ineq} \) and \( \mathbf{L}_{eq} \) are the Lagrange multipliers for the linear inequality and equality constraints.

The most widely used relaxation approaches are reviewed below.

**Spectral Relaxation:** This relaxation is originally designed for U.BQP problem. The binary constraint is relaxed to a spherical constraint.

\[
\begin{align*}
\text{(Spectral)} & \quad \max_{\mathbf{v}} \mathbf{v}^T \mathbf{Qv}, \text{s.t. } \|\mathbf{v}\|_2 = 1.
\end{align*}
\]

The binary solution is then obtained by \( \mathbf{s} = \text{sign}(\mathbf{v}_{\text{max}}) \), where \( \mathbf{v}_{\text{max}} \) is the leading eigenvector of \( \mathbf{Q} \). The computation of this approach is simple and is extremely fast. However, the solution quality is very limited.

**Semi-definite Relaxation (SDR):** The binary constraint is relaxed to a semidefinite constraint [32,35,42]

\[
\begin{align*}
\text{(SDR)} & \quad \min_{\mathbf{s}, \mathbf{Q}} < \mathbf{Q}, \mathbf{s} >, \text{s.t. } \mathbf{s} \in \{\pm 1\}^n \text{ and } \|\mathbf{AS} - \mathbf{b}\|_2 = 0.
\end{align*}
\]

for some matrix \( \mathbf{A} \) and vector \( \mathbf{b} \). It is noted that SDR is designed for linear equality constraints and quadratic constraints. SDR outputs a semidefinite matrix \( \mathbf{S} \) as the optimal solution. The binary solution is then obtained by applying the random hyperplane rounding technique to the matrix \( \mathbf{S} \) [30]. The solution quality of this approach is currently the highest owing to its tight bound formulation. However, the computation of the semidefinite matrix always leads to high complexity. Besides, the random hyperplane rounding technique adopts an independent random search. A large number of trials is needed in order to obtain a high-quality solution.

**Exact Penalty Method (EPM)** The binary constraint is relaxed to a non-separable \( \ell_2 \) box [41]

\[
\begin{align*}
\text{(EPM)} & \quad \min_{\mathbf{s}, \mathbf{v}} s^T \mathbf{Qs}, \text{s.t. } \mathbf{s} \in [-1,1]^n, s^T \mathbf{v} = n, \\
& \quad \|\mathbf{v}\|_2 \leq n, \mathbf{s} \in \Omega.
\end{align*}
\]

The introduction of the variable \( \mathbf{v} \) turns the BQP problem to a biconvex problem. This can then be solved by standard optimization techniques. However, the solution quality may not be as high as SDR, owing to the fact that a biconvex problem is usually not convex. This makes the obtained solution optimal only in a local convex ball.

3 PROPOSED SOLVER FOR G.BQP PROBLEMS

We solve the G.BQP problems by characterizing the locally optimal solution \( s^* \) over its stochastic neighborhood. We propose a new algorithm so that the locally optimal solution \( s^* \) possesses the following property

\[
\begin{align*}
\mathbf{f}(s^*) \geq \mathbf{f}(\mathbf{s}), \forall \mathbf{s} \in \text{Nbd}(s^*).
\end{align*}
\]

where \( \text{Nbd}(s^*) \) is a stochastic neighborhood of \( s^* \). The key idea of the proposed method is to perform a large perturbation to a potential optimal solution and obtain a stochastic neighborhood vector of this potential solution. We then apply a steepest ascent search (this will be explained in Section 3.1) and check if this can output a better objective function value. If this is the case, we adopt this output as a new potential solution. We then perturbate this new solution.
and find a new stochastic neighborhood vector (this will be explained in Section 3.2). These two steps are repeated a number of times. This is illustrated in Figure 1. The merit of the proposed method is that starting from a lower quality solution, the perturbation can introduce a stochastic neighborhood vector that eventually leads to a better solution. The design of the proposed solver consists of two main parts: (i) Deterministic steepest ascent search; (ii) Finding a set of stochastic neighborhood vectors of the optimal solution.

3.1 Deterministic Steepest Ascent Search

We adopt a projected gradient ascent method to find a local optimal solution of the G.BQP problem that iteratively improve the solution quality. To update the binary vector at the \((t-1)\)th iteration \(s_{t-1}\), we first find an intermediate vector by solving a dual convex optimization problem (ALGO 1). The optimal step size that can speed up the convergence of ALGO 1 is introduced in ALGO 2. Finally, we repeat ALGO 1 several times and obtain several intermediate vectors. The one which can best improve the solution is the updated vector \(s_t\). This is ALGO 3.

The updated vector can be obtained by solving the following optimization problem

\[
s_t = \arg\min_{s \in \{-1,1\}^n} \|s - (s_{t-1} + \alpha'Qs_{t-1})\|_2^2 \quad \text{s.t.} \quad A_{ineq}s \leq b_{ineq} \quad \text{and} \quad A_eqs = b_eq.
\]

The vector \(Qs_{t-1}\) can be treated as the gradient of the primal function \((s^TQs)\) and thus the ascent direction. \(\alpha'\) is the step size. It is noted that \(s \in \{-1,1\}^n\). By expressing the squared \(L_2\) norm of Equation (3), the problem can be rewritten as

\[
s_t = \arg\max_{s \in \{-1,1\}^n} s^T(s_{t-1} + \alpha'Qs_{t-1}),
\]

\[
\text{s.t.} \quad A_{ineq}s \leq b_{ineq} \quad \text{and} \quad A_eqs = b_eq.
\]

Since a constant multiplication to the objective function does not have any impact to the optimal solution, Equation (4) can be expressed as

\[
s_t = \arg\max_{s \in \{-1,1\}^n} s^Tq(s_{t-1}),
\]

\[
\text{s.t.} \quad A_{ineq}s \leq b_{ineq} \quad \text{and} \quad A_eqs = b_eq,
\]

where \(q(s_{t-1}) = (1 - \alpha) s_{t-1} + \alpha Qs_{t-1}, \alpha \in (0,1)\). This is obtained by taking \(\alpha' = \alpha/(1 - \alpha), \forall \alpha \in (0,1)\) in Equation (4). The greatest merit of this expression over the original one is that the range of \(\alpha'\) is from zero to infinity whereas \(\alpha\) is in a bounded domain.

Equation (5) can be converted into a convex optimization problem. By introducing the Lagrange multipliers \(L_{ineq}\) and \(L_eq\), we obtain the dual form of Equation (5) as below

\[
\min_{L_eq,L_{ineq} \geq 0} \max_{s \in \{-1,1\}^n} s^Tq(s, s_{t-1}) + L_{ineq}(A_{ineq}s - b_{ineq}) + L_eq(A_eqs - b_eq)
\]

\[
= \min_{L_eq,L_{ineq} \geq 0} \max_{s \in \{-1,1\}^n} s^T[q(s, s_{t-1}) + A_{ineq}^T L_{ineq} + A_eq^T L_eq]
\]

\[
= \|q(s, s_{t-1}) + A_{ineq}^T L_{ineq} + A_eq^T L_eq\|_1
\]

\[
\|L_{ineq} b_{ineq} + L_eq b_{eq}\|
\]

If we consider the maximization part of the problem, it can be simplified as

\[
\max_{s \in \{-1,1\}^n} s^T[q(s, s_{t-1}) + A_{ineq}^T L_{ineq} + A_eq^T L_eq]
\]

\[
- \|L_{ineq} b_{ineq} + L_eq b_{eq}\|
\]

Thus, the dual problem becomes

\[
\min_{L_eq,L_{ineq} \geq 0} f(L_eq, L_{ineq})
\]

\[
= \min_{L_eq,L_{ineq} \geq 0} [\|q(s, s_{t-1}) + A_{ineq}^T L_{ineq} + A_eq^T L_eq\|_1]
\]

\[
- \|L_{ineq} b_{ineq} + L_eq b_{eq}\|
\]

This is obviously a convex optimization problem and the global optimal solution of the pair \(L_eq\) and \(L_{ineq}\) can be obtained. With the above dual problem, we can obtain the updated binary vector \(s_t\) by the following formula

\[
s_t = \text{sign}(q(s, s_{t-1}) + A_{ineq}^T L_{ineq} + A_eq^T L_eq)
\]

This formula is obtained based on the fact that the optimal solution \(s\) of the problem \(s = \arg \max_{s \in \{-1,1\}^n} s^T x\) is \(s = \text{sign}(x)\). Here, \(x = [q(s, s_{t-1}) + A_{ineq}^T L_{ineq} + A_eq^T L_eq]\).

To find the optimal solution of Equation (8), we adopt the steepest descent method. The algorithm is shown in ALGO 1 (Solve_Dual). In the initialization step, we select \(L_{ineq,0} = 0\) and \(L_eq,0 = 0\), which belong to the feasible region of the dual problem (\(L_{ineq} \leq 0\) and \(L_eq \leq \text{any real vector}\)). Next, we find the negative gradient directions of the function \(f(L_eq, L_{ineq})\) with respect to \(L_eq\) and \(L_{ineq}\). These are Steps 1 and 2. After that, we find the optimal step size for the descent direction, i.e. we adjust the step size of the descent direction that can best improve the function \(f(L_eq, L_{ineq})\). This involves solving the following optimization problem

\[
\min_{\gamma} \phi(\gamma) = \min_{\gamma} f(L_eq,k-1 + \gamma dL_eq, L_{ineq},k-1 + \gamma dL_{ineq})
\]

(10)

This is Step 3 and will be explained later. Step 4 is to update the variable pair \((L_eq, L_{ineq})\). In order to satisfy the constraint \(L_{ineq} \leq 0\), we impose the condition \(\min(\gamma dL_{ineq}, 0)\) in the updating step for the inequality constraint. This condition ensures that \(L_{ineq,k} \leq 0\) because \(L_{ineq,k-1} \leq 0\) and the quantity \(\min(\gamma dL_{ineq}, 0)\) is at most zero (Note: \(\gamma \geq 0\)). Note that upon convergence, we must have \(dL_{ineq} \geq 0\). If
not, $L_{ineq,k}$ can still be improved. Thus, this implies $0 \leq dL_{ineq} = b_{ineq} - A_{ineq} \mathbf{s}$. In other words, the linear inequality constraints of the primal problem are satisfied. This will be used as one of the clues to prove Proposition 1 below. In Step 5, the algorithm goes back to Step 1 if not converged, otherwise, the solution to Equation (8) is outputted.

**Input:** $q(\alpha, s_{t-1}), A_{ineq} A_{eq}, b_{ineq}, b_{eq}$

**Output:** $s, L_{eq}$ and $L_{ineq}$

Step 0: Set $k = 0$, $L_{ineq,0} = 0$ and $L_{eq,0} = 0$

Step 1: Compute $s = \text{sign}(q(\alpha, s_{t-1}) + A_{eq} L_{ineq,k} + A_{eq} L_{eq,k})$ [Equation 9]

Step 2: Compute $dL_{ineq} = b_{eq} - A_{ineq} s$

and $dL_{eq} = b_{eq} - A_{eq} s$

Step 3: Compute $\gamma = \arg \min \phi(\gamma)$. [ALGO 2 (Find Step Size)], $\gamma = \max(\gamma, 0)$

Step 4: Compute $L_{ineq,k} = L_{ineq,k-1} + \min(\gamma dL_{ineq,0})$ and $L_{eq,k} = L_{eq,k-1} + \gamma dL_{eq}$

Step 5: If $\left(\left|L_{ineq,k} - L_{ineq,k-1}\right| + \gamma dL_{ineq,0}\left|L_{eq,k-1}\right|\right) \leq \varepsilon$

output $s, L_{eq} = L_{eq,k}$ and $L_{ineq} = L_{ineq,k}$; Otherwise, take $k = k + 1$ and go to Step 1.

**ALGO 1. Optimization Procedure for the Dual Problem (Solve_Dual).**

Now, we explain the details of Step 3 of ALGO 1. By substituting $L_{eq} = L_{eq,k-1} + \gamma dL_{eq}$ and $L_{ineq} = L_{ineq,k-1} + \gamma dL_{ineq}$ to Equation (8), Equation (10) can be rewritten as

$$\min_{\gamma} \phi(\gamma) = \min \left\{ \frac{\beta}{\gamma} + \gamma \beta, \gamma \right\},$$

where

$$\beta = q(\alpha, s_{t-1}) + A_{eq} L_{ineq,k-1} + A_{eq} L_{eq,k-1}, \beta = dL_{eq} + dL_{ineq}$$

This is also a convex optimization problem and the global optimal solution $\gamma$ can be obtained with the aid of the following theorem.

**Theorem 1:** Consider the following optimization problem,

$$\min_{\gamma} \phi(\gamma) = \min \left\{ \frac{\beta}{\gamma} + \gamma \beta, \gamma \right\},$$

Without loss of generally, we assume that (i) there are non-zero $\hat{d}_i$ and (ii) $0 \leq \frac{\hat{d}_i}{d_i} \leq \frac{\hat{d}_i}{d_i}$, $0 \leq \frac{\hat{d}_i}{d_i} \leq \frac{\hat{d}_i}{d_i}$. The global optimal solution of Equation (12) is attained at $\gamma^* = \frac{\varepsilon}{\varepsilon}$ with index $i$, minimizing the equation $i = \arg \min_{i} \beta_{eq} + 2 \Sigma_{i=1}^{n} d_i$.

**Proof:** As the function $\phi(\gamma)$ is convex, the zero of its first derivative has a closed-form solution. The proof is shown in Appendix MP.

With Theorem 1, the algorithm that solves Equation (12) is shown in ALGO 2 (Find Step Size). In Step 2 of ALGO 2, the minimization problem for the index $i$, can be quickly solved. This is because the two terms $\Sigma_{i=1}^{n} d_i$ and $\Sigma_{i=1}^{n} d_i$ only involves summation of $\left|\hat{d}_i\right|$. The computation complexities are very low. Next, we prove that ALGO 1 (Solve_Dual) converges.

**Theorem 2:** The iterative algorithm ALGO 1 (Solve_Dual) converges.

**Proof:** We show that the gradient direction is guaranteed to be a decent direction, i.e., $f(L_{eq,k+1}, L_{ineq,k+1}) \leq f(L_{eq,k}, L_{ineq,k})$, and it iteratively improves the solution. Thus, the algorithm converges. The proof is shown in Appendix MP.

**Input:** $\mathbf{q}, \mathbf{d}, \mathbf{b}$

**Output:** $\gamma$

Step 0: Consider the pairs $(\hat{q}_1, \hat{d}_1), \ldots, (\hat{q}_n, \hat{d}_n)$, with each of them has non-zero $\hat{d}_i$

Step 1: Sort the sequence so that $\frac{\hat{q}_1}{\hat{d}_1} \leq \ldots \leq \frac{\hat{q}_n}{\hat{d}_n}$

Step 2: Set $\gamma = \frac{\hat{q}_1}{\hat{d}_1}$, with index $i$, minimizing the equation $\alpha_i = \arg \min_{i} \left|b - \sum_{i=1}^{n} \hat{d}_i + 2 \sum_{i=1}^{n} \hat{d}_i\right|$

**ALGO 2. Finding Optimal Step Size of Descent Direction (Find Step Size).**

**Input:** $s_{d}, Q, A_{ineq} A_{eq} b_{ineq}, b_{eq}$

**Output:** $s, L_{eq}$ and $L_{ineq}$

Step 0: Initialization $t = 0$

Step 1: $\alpha = 1/5, 2/5, \ldots, 1$

Step 1(i): Compute $\mathbf{q}_1 = \mathbf{Q} s_{d}, \mathbf{q}_2 = \mathbf{Q}, c_{nor} = \|\mathbf{q}_2\|_2/\|\mathbf{q}_1\|_2, \mathbf{q}_1 = c_{nor} \mathbf{q}_1, \mathbf{q} = \alpha \mathbf{q}_1 + (1 - \alpha) \mathbf{q}_2$

Step 1(ii): Apply ALGO 1 (Solve_Dual) with $q(\alpha, s_{t-1}) = \mathbf{q}, A_{ineq} A_{eq} b_{ineq}, b_{eq}$ and obtain the output $s_{t,eq}, L_{eq,t}, L_{ineq,t}$. Compute $obj(\alpha) = s^T Q s$

Step 2: $\alpha^* = \max_{\alpha} obj(\alpha)$. Set $\mathbf{s}_{t} = \mathbf{s}_{t,eq}, L_{eq,t} = L_{eq,eq}, c_{nor} = L_{ineq,t} = L_{ineq,eq}$

Step 3: If $s_t = s_{t-1}, t = t + 1$ and go to Step 1. Otherwise, output $s = s_t, L_{eq} = L_{eq,t}/c_{nor}$ and $L_{ineq} = L_{ineq,t}/c_{nor}$

**ALGO 3. Deterministic Ascent Search (Ascent_Search).**

**Corollary 1:** The rate of convergence $\rho$ of the iterative algorithm ALGO 1 (Solve_Dual) defined as $\rho = \frac{f(L_{eq,k+1}, L_{ineq,k+1}) - f(L_{eq,k}, L_{ineq,k})}{f(L_{eq,k+1}, L_{ineq,k+1}) - f(L_{eq,k}, L_{ineq,k})}$, where $L_{eq}$ and $L_{ineq}$ are the optimal solutions, is at least linear.

**Proof:** This is based on the result of Theorem 2. The proof is shown in Appendix MP and validated in Appendix CV.

ALGO 3 (Ascent_Search) shows the complete algorithm for the deterministic ascent search. In Step 1(i) of ALGO 3, we normalize the vector $\mathbf{q}_1$ so that its strength is roughly the same as $\mathbf{q}_2$. This makes the two vectors $\mathbf{q}_1$ and $\mathbf{q}_2$ not...
dominating each other when combined. The Lagrange multipliers \( L_{\text{eq}} \) and \( L_{\text{ineq}} \) are obtained by reverting the normalization, which is given by \( L_{\text{eq}} = L_{\text{eq}}/c_{\text{nor}} \) and \( L_{\text{ineq}} = L_{\text{ineq}}/c_{\text{nor}} \) (Step 3). In Step 2, we update the binary vector \( s_{t-1} \) by ALGO 1 (Solve_Dual) and select the optimal step size \( \alpha \) that maximizes the primal object function \( s^T \tilde{Q} s \).

Next, we are going to prove that ALGO 3 (the iterative procedure of solving Equation (9)) converges. Moreover, it increases the value of the primal function \( (s^T \tilde{Q} s) \) iteratively and eventually satisfies all the KKT optimality conditions (Equation (1)). But before showing the proof, we need the following proposition.

**Proposition 1:** The optimal solution \( L_{\text{ineq}} \) of the function \( f(L_{\text{eq}}, L_{\text{ineq}}) \) satisfies the KKT-complementary slackness condition (Equation (1)). That is, \( L_{\text{ineq}} \cdot \left( A_{\text{ineq}} s - b_{\text{ineq}} \right) = 0 \).

**Proof:** If \( L_{\text{ineq}} \cdot \left( A_{\text{ineq}} s - b_{\text{ineq}} \right) \neq 0 \), the solution is not optimal. The proof is shown in Appendix MP.

**Theorem 3:** There exists \( \alpha > 0 \) such that the iterative procedure

\[
s_t = \text{sign}(q(\alpha, s_{t-1}) + A_{\text{eq}}^T L_{\text{eq}, t} + A_{\text{ineq}}^T L_{\text{ineq}, t} )
\]

implies

\[
s_t^T Q s_t \geq s_{t-1}^T Q s_{t-1},
\]

\( \forall t \geq 0 \). Moreover, \( s_{t-1} = \text{sign}(a Q s_{t-1} + A_{\text{eq}}^T L_{\text{eq}, t-1} + A_{\text{ineq}}^T L_{\text{ineq}, t-1} ) \) if and only if \( s_t = s_{t-1} \) and \( s_t \) satisfies the KKT optimality conditions as stated in Equation (1). In other words, the proposed solver converges to a locally optimal solution.

**Proof:** The first half is proved by the trick \( \exists \alpha > 0 \) such that \( 0 \leq (s_t - s_{t-1})^T (a Q + (1 - \alpha) I)(s_t - s_{t-1}) \). The second half is proved by checking the conditions stated in Equation (1). The proof is shown in Appendix MP.

### 3.2 Finding a Set of Stochastic Neighborhood Vectors in the Optimal Solution

We first apply bootstrapping to the G.BQP problem and obtain a neighborhood function. We then find a neighborhood vector by applying the projected gradient ascent method.

#### 3.2.1 Applying bootstrapping to the G.BQP problem

The Lag.GBQP problem is

\[
\text{(Lag,GBQP)} \quad \min_L \max_{\alpha \in [-1,1]^n} s^T Q s + L^T (A s - b)
\]

where \( L = \left[ L_{\text{eq}} \right] \) (\( L_{\text{eq}} \leq 0 \)), \( A = \left[ A_{\text{eq}} \right] \) and \( b = \left[ b_{\text{ineq}} \right] \).

This formulation can be expressed as a function of \( B(s, Q, L) \) that can be viewed as the average of certain second central moment functions \( B_i(s, \tilde{x}, \tilde{y}, Q, L) \)

\[
B(s, Q, L) = \frac{1}{n} \sum_{i=1}^{n} B_i(s, \tilde{x}, \tilde{y}, Q, L) + \frac{(L^T b)}{n},
\]

where \( B_i(s, \tilde{x}, \tilde{y}, Q, L) = n\|s_i \tilde{x}_i - \tilde{y}_i \|^2 - n\|s_i \tilde{y}_i - \tilde{y}_i \|^2 + s_i (L^T A)_{ij} \) with \( \tilde{x} = \frac{1}{n} \sum_{j=1}^{n} s_j \tilde{x}_j \) and \( \tilde{y} = \frac{1}{n} \sum_{j=1}^{n} s_j \tilde{y}_j \). This is justified in the following theorem.

**Theorem 4:** \( B(s, Q, L) \) is proportional to the Lag.GBQP problem.

**Proof:** The result is obtained by expressing \( B(s, Q, L) \) in quadratic form with matrix notation. The proof is shown in Appendix MP and validated in Appendix MP-VT.

The idea of bootstrapping is to apply random sampling with replacement to the collected samples and estimate the distribution of a statistic such as the statistical mean \([43, 44]\). In our case, we view \( B(s, Q, L) \) as an average of the terms \( B_i(s, \tilde{x}, \tilde{y}, Q, L) \). We find the distribution of this average by bootstrapping these terms. Statistically, this distribution lists all possible outcomes and they are the stochastic neighborhood of the average. To implement this concept, we follow the standard bootstrapping procedure \([45]\) and randomly select the terms \( B_i(s, \tilde{x}, \tilde{y}, Q, L) \) with replacement \( n \) times. This gives the bootstrapped function

\[
B_b(s, Q, L) = \frac{1}{n} \sum_{i=1}^{n} w_i B_i(s, \tilde{x}, \tilde{y}, Q, L) + \left( \frac{L^T b}{n} \right),
\]

where \( w_i \) is the number of times the term \( B_i(s, \tilde{x}, \tilde{y}, Q, L) \) is sampled. Here, \( B_b(s, Q, L) \) is a stochastic neighborhood function of \( B(s, Q, L) \), and hence also of Lag.GBQP problem. The stochastic neighborhood function can be re-stated in a matrix quadratic form. By expressing the squared \( l_2 \) norms of \( B(s, \tilde{x}, \tilde{y}, Q, L) \) in matrix notation, the stochastic neighborhood function can be written as

\[
\max_{s \in \{ -1, 0, 1 \}^n} s^T W Q W s + L^T (A W s - b)
\]

\( W \) is a diagonal matrix with \( W_{ij} = w_i \). It is noted that there are three possible values, i.e. \(-1, 0, 1\), for the elements of \( s \), as some element \( w_i \) is zero and we set the corresponding elements of \( s \) as zeros. In contrast with the Lag.GBQP problem, we have introduced the diagonal matrix \( W \) to the vector \( s \).

#### 3.2.2 Obtaining a neighborhood vector

To obtain the neighborhood vector \( s_{\text{local}} \) of the local optimal solution \( s_{\text{local}} \), we apply the projected gradient ascent method (introduced in Section 3.1) to the primal objective function. (i.e., \( s^T W Q W s \)) and obtain the following optimization problem.

\[
s_{\text{local}} = \arg \max_{s \in \{ -1, 0, 1 \}^n} s^T (W_{\alpha, s_{\text{local}}}), \text{ s.t. } A W s = b.
\]

where \( W_{\alpha, s_{\text{local}}} = \alpha W_{s_{\text{local}}} + (1 - \alpha) W Q W \).

\( \Lambda(W) \) is a diagonal matrix with \( \Lambda(W)_{ii} = 0 \) if \( W_{ii} \neq 0 \). Otherwise, \( \Lambda(W)_{ii} = 0 \). To simplify the calculation, we replace the constraint \( A W s = b \) by \( 1^T W s = \tau \), where \( w_i = 1 \) if \( w_i \neq 0 \). Otherwise, \( w_i = 0 \). \( \tau \) is any random integer between \(-1^T W 1_w \) and \( 1^T W 1_w \). \( 1_w \) represents the number of non-zero elements of \( W \). This simplification has a
similar meaning to the original constraint \( AS = b \). The original constraint requires that the weighted sum of \( Ws \) equals to \( b \) while this random constraint requires that the weighted sum of \( Ws \) equals to a random number. Moreover, this constraint allows the three possible values \([-1, 0, 1]\) of \( s \) to be of any ratio, which means the stochastic neighborhood vector \( s \) generated each time can be very different. A larger stochastic neighborhood range means the solution is optimal in a larger region and thus, is closer to a better solution. Now, we show the problem with the simplified constraint below:

\[
\max_{s \in \{-1, 0, 1\}^n} s^T (qW(a, s_{\text{local}})) s + 1^T s = \tau. \tag{19}
\]

By introducing the Lagrange multiplier \( R_{rn} \), the dual form of this problem is

\[
\begin{align*}
\min_{R_{rn}} \max_{s \in \{-1, 0, 1\}^n} & s^T qW(a, s_{\text{local}}) + R_{rn}(s^T 1_w - \tau) \\
= & \min_{R_{rn}} \max_{s \in \{-1, 0, 1\}^n} s^T (qW(a, s_{\text{local}}) + R_{rn} 1_w) - R_{rn} \tau \\
= & \min_{R_{rn}} \max_{s \in \{-1, 0, 1\}^n} \|qW(a, s_{\text{local}}) + R_{rn} 1_w\|_1 - R_{rn} \tau.
\end{align*} \tag{20}
\]

Equation (20) is the same as Equation (12). Thus, using the result of Theorem 1 and ALGO 2 (Find_StepSize), the global optimal solution of the above problem can be obtained. After finding \( R_{rn} \), the stochastic neighborhood vector \( s_{\text{nbd}} \) can then be obtained by

\[
s_{\text{nbd}} = \text{sign}(qW(a, s_{\text{local}}) + R_{rn} 1_w). \tag{21}
\]

The following experiment further illustrates the impact of this random constraint. We apply the solutions of Equation (20) with \( R_{rn} = 0 \) (without the constraint) and Equation (20) (with the constraint) to a U.BQP problem with \( Q \in \mathbb{R}^{100 \times 100} \). The resulting projections (i.e. \( Qs \in \mathbb{R}^{100} \)) with 500 runs are summarized in Figure 2. The three lines shown in the figures indicate the minimum (the lower one), the median (the middle one) and the maximum (the upper one) of the 500 runs. It is clearly shown that the variation of the projection with the random constraint is higher than that without the random constraint.

![Illustration of the stochastic neighborhood vectors with and without random constraints.](image)

ALGO 4 summarizes the above procedure that outputs a stochastic neighborhood vector \( s_{\text{nbd}} \) of the locally optimal solution \( s_{\text{local}} \). First, we generate a set of random integers and obtain the weights \( w_i \) for bootstrapping. Similar to ALGO 3 (Ascent_Search), we apply normalization to \( q_1 \), so that its strength is similar to \( q_2 \). We then apply ALGO 2 (Find_StepSize) and obtain the stochastic neighborhood vector \( s_{\text{nbd}} \). In step 2, the step size \( \alpha \) is selected as any random number from \([0, 1]\). This means that the generated stochastic neighborhood vector can be closed (when \( \alpha \to 0 \)) to the local optimal solution \( s_{\text{local}} \) or far away (when \( \alpha \to 1 \)) from the local optimal solution \( s_{\text{local}} \). Lastly, to enhance the quality of the stochastic neighborhood vector \( s_{\text{nbd}} \), we flip the vector by \( s_{\text{nbd}} = -s_{\text{nbd}} \) (Step 3) if \( L^T AWs_{\text{nbd}} < 0 \). This condition is obtained from the objective function \( s^T WQW + L^T AW + L^T b \) (Equation (17)). If \( L^T AWs_{\text{nbd}} < 0 \), the objective function value is smaller. Thus, flipping the sign of \( s_{\text{nbd}} \) leads to a better stochastic neighborhood vector.

**Input:** \( s_{\text{local}}, Q, L, A \)

**Output:** \( s_{\text{nbd}} \)

**Step 1:** Generate \( n \) random integers with range \([1, n]\). Obtain the diagonal matrix \( W \) with diagonal entries \( W_{ii} = \) the number of times the integer \( i \) appears.

**Step 2:** Select a uniform random number from \([0, 1]\) and assign it to \( \alpha \)

(i) Set \( \Lambda(W) \) as a diagonal matrix with \( (\Lambda(W))_{ii} = 1 \) if \( W_{ii} \neq 0 \). Otherwise, \( (\Lambda(W))_{ii} = 0 \).

(ii) Compute \( q_1 = WQW_{\text{local}}q_2 = \Lambda(W)s_{\text{local}}q_1 = \|q_2\|_2 q_1 / \|q_2\|_2 \).

(iii) Compute \( q_0 = \alpha q_1 + (1 - \alpha) q_2 \).

(iv) Count the number of non-zero elements of \( WS \) and compute \( -1^T W_{\text{nbd}} \) and \( 1^T W_{\text{nbd}} \).

(v) Randomly select an integer between \( -1^T W_{\text{nbd}} \) and \( 1^T W_{\text{nbd}} \) and assign it to \( \tau \).

(vi) Apply ALGO 2 (Find_StepSize) with \( \hat{q} = q_0, \hat{d} = 1, \hat{b} = \tau \) and output \( R_{rn} = \gamma \).

(vii) Obtain the vector \( s_{\text{nbd}} = \text{sign}(q_0 + R_{rn} 1_w) \) (Equation (21)).

**Step 3:** If \( L^T AWs_{\text{nbd}} < 0 \), \( s_{\text{nbd}} = -s_{\text{nbd}} \).

**ALGO 4:** Procedure for Obtaining a stochastic neighborhood vector from \( s_{\text{local}} \) (Get_NV).

### 3.3 Algorithm for the Proposed Solver

In this section, we combine the above procedures and present the complete algorithm. We also apply the Laplace normalization technique to generate an initial guess, which will be mathematically shown to be more effective than the spectral relaxation method.

#### 3.3.1 Complete Algorithm of the Proposed Solver

The complete algorithm is summarized in ALGO 5. We first explain Step 0(iii), Step 1 and 2. The initialization step (Step 0 (ii)) will be explained later. For Step 0(iii), it is to set the diagonal of the matrix \( Q \) to be zeros. This setting is to remove the biased in updating the binary vector. In Equation (9), \( q(a, s_{t-1}) = \alpha s_{t-1} + (1 - \alpha) Qs_{t-1} \). If the diagonal entries of \( Q \) are not zero, it will make extra-contribution to the term \( \alpha s_{t-1} \). In Step 1, we apply the projected gradient...
ascent method (ALGO 1 (Solve_Dual)) to obtain a local optimal solution \(s^*\). In Step 2, a stochastic neighborhood vector of this local optimal solution is then generated using Get_NV (ALGO 4). If the obtained stochastic neighborhood vector produces a larger objective function value (i.e., \(J > J_{\text{max}}\)), we apply the projected gradient ascent method (ALGO 1) again, i.e., go to Step 1. Otherwise (i.e., \(J \leq J_{\text{max}}\)), we find another stochastic neighborhood vector by applying Get_NV (ALGO 3). This procedure is repeated until we find an optimal solution \(s^*\), which is better than \(m = 50\) stochastic neighborhood vectors.

**Step 0: Initialization**

- **Step 0(i):** Set \(l = 1\) and \(m = 50\).
- **Step 0(ii):** Obtain \(s_{\text{local}} = v\) by the procedure shown in ALGO 6. Set \(J_{\text{max}} = -\infty\).

**Step 0(iii):** Set \(Q_0 = 0\). 

**Step 1: Ascent Search:** Obtain \(s_{\text{local}}, L_{\text{ineq}}\) and \(L_{\text{eq}}\) using Ascent_search (ALGO 3) with input \(s_{\text{local}}, Q, A_{\text{ineq}}, A_{\text{eq}}, b_{\text{ineq}}, b_{\text{eq}}\). Compute \(J = s_{\text{local}}^T Q s_{\text{local}}\).

- If \(J > J_{\text{max}}\), set \(l = 1, s^* = s_{\text{local}}, L^* = \left[\begin{array}{c|c} L_{\text{ineq}} & 0 \\ \hline 0 & L_{\text{eq}} \end{array}\right]\) and \(J_{\text{max}} = J\).

**Step 2: Neighborhood Projection:** Set \(l = l + 1\). If \(l \leq m\),

- **Step 2(i):** Obtain \(s_{\text{local}}\) using Get_NV (ALGO 4) with input \(s^*, L^*, Q, A = \left[\begin{array}{c} A_{\text{ineq}} \\ A_{\text{eq}} \end{array}\right]\).
- **Step 2(ii):** Go to Step 1.

Otherwise (i.e., \(l > m\)), output the optimal solution \(s^*\).

ALGO 5. Proposed Stochastic Solver for the BQP Problem.

Our strategy for the initialization (Step 0 of ALGO 5) of the stochastic solver is to express the Lag-GBQP problem into a quadratic form below

\[
\max_{x, |x|_2=1} x^T P x.
\]

The vector \(x\), which is the eigenvector corresponding to the largest eigenvalue of \(P\), is used as an initial guess to ALGO 5. To achieve this, we first express the Lag-BQP problem into the quadratic form \(x^T P x\). Then, we relax the binary constraints \(s \in \{-1,1\}^n\) and estimate it using the spherical constraint \(|x|_2 = 1\) by applying the Laplace normalization.

3.3.3 Expressing the Lag-BQP Problem in Quadratic Form

The Lag-BQP problem can be written as

\[
\max_{s \in \{-1,1\}^n} s^T Q s.
\]

The vector \(x\), which is the eigenvector corresponding to the largest eigenvalue of \(P\), is used as an initial guess to ALGO 5. To achieve this, we first express the Lag-BQP problem into the quadratic form \(x^T P x\). Then, we relax the binary constraints \(s \in \{-1,1\}^n\) and estimate it using the spherical constraint \(|x|_2 = 1\) by applying the Laplace normalization.

Obviously, if \(\lambda_{\text{GBQP}}\) is large enough, the second and third terms will go to zero and the linear equality constraints in Equation (23) are satisfied. However, if \(\lambda_{\text{GBQP}}\) is too large, it will dominate the whole function and the objective of maximizing \(s^T Q s\) cannot be achieved. So, we impose a condition below to estimate the value of \(\lambda_{\text{GBQP}}\).

\[
|s^T Q s| \geq \lambda_{\text{GBQP}} \left[ \|A_{\text{eq}} s - b_{\text{eq}}\|^2 + \|A_{\text{ineq}} s + d - b_{\text{ineq}}\|^2 \right]
\]

or

\[
|s^T Q s| \left[ \|A_{\text{eq}} s - b_{\text{eq}}\|^2 + \|A_{\text{ineq}} s + d - b_{\text{ineq}}\|^2 \right] \geq \lambda_{\text{GBQP}}
\]

The inequality can ensure that the term \(|s^T Q s|\) has a larger contribution. Since \(|s^T Q s| \leq \|Q\|_1\), we only need to estimate \(s\) and \(d\). We first adopt the spectral method to estimate the binary vector \(s\). That is, \(s = \text{sign}(v_{\text{max}})\), where \(v_{\text{max}} = \arg\max_{v \in \mathbb{R}^n} v^T Q v\). By Equation (23), the slack variable \(d\) can be estimated as \(d = \max(b_{\text{ineq}} - A_{\text{ineq}} \text{sign}(v), 0)\). With these estimates, \(\lambda_{\text{GBQP}}\) is taken as the upper bound of Equation (25) and Equation (24) can be expressed as

\[
\max_{s_0, s = \{-1,1\}^n} \left( s_0 s^T \right) \left[ G |s|_2 \right].
\]

It is noted that in order to convert Equation (23) to Equation (26), we introduce a dummy variable \(s_0\). This conversion is valid only if \(s_0 = 1\). Thus, if \(s_0 = -1\), we can multiply the whole vector \(|s|^2\) by -1 and enforce \(s_0 = 1\). Owing to the symmetric property of the function, the multiplication of -1 does not change the objective function value.

3.3.3 Relaxing the Binary Constraint

Now, we apply Lagrangian method to relax the binary constraint. It is obvious that Equation (26) is a U.BQP problem. The Lagrange formulation of U.BQP that turn all binary constraints into the Lagrange form is

\[
\min \max_{D} s^T (P - D) s + 1^T D 1,
\]

where \(D\) is a diagonal matrix and its diagonal entries are the Lagrange multipliers for the constraints \(s_0 = 1\). This formulation relaxes the binary vector \(s\) to be a continuous vector. If \(s^T (P - D) s\) is non-positive, \(\max{s} s^T (P - D) s\) is bounded above by zero. If \(s^T (P - D) s\) is positive, \(\max{s} s^T (P - D) s\) goes to infinity. This means the Lag-UBQP problem is valid only if \(s^T (P - D) s \leq 0\) or \(s^T P s \leq s^T D s\). In other words, \(s^T (P - D) s \leq 0\) is a constraint. This implies that the dual formulation is

\[
\min \max_{D} s^T (P - D) s + 1^T D 1
\]

\[
= \begin{cases} 
    \min 1^T D 1 & \text{if } s^T (P - D) s \leq 0 \forall s \\
    \infty & \text{otherwise}
\end{cases}
\]

In this dual formulation, we can see that a feasible solution \(D\) (i.e., \(s^T P s \leq s^T D s\)) that gives a smaller objective function
value (i.e. $1^T D1$) provide a tighter bound to the primal solution. In other words, if $1^T D1$ is smaller, $s^T D s$ will approach $s^T P s$ and thus closer to the optimal solution. We now show that the solution estimated by the Laplace normalization technique gives a tighter solution bounded than that of the spectral relaxation.

For the spectral relaxation, we put $D = \lambda_{\text{max}} P$ to Equation (28), where $\lambda_{\text{max}}$ is the maximum eigenvalue of $P$. Since $P_{ij} = 0 \forall i$, the maximum eigenvalue must be nonnegative. This means the constraint is satisfied. That is, $s^T P s \leq s^T D s = \lambda_{\text{max}} s^T s$. The objective function of the dual problem (Equation (28)) is then $1^T D1 = 1^T (\lambda_{\text{max}} P)1 = n \lambda_{\text{max}} P$.

Here, we apply the Laplacian normalization technique to estimate the Lagrange multipliers $D$ that yield a better estimate than the spectral relaxation method. The Laplacian normalization technique has been widely used in the signed graph modelling [46]. Here, we briefly review the relevant theory. The signed graph model is

$$s^T (P - D) s, \quad \text{maximize}$$

where $D$ is a diagonal matrix with $D_{ii} = \sum |P_{ij}|$. The normalized Laplacian transforms the problem by taking $g_{\text{Lap}} = D^{-\frac{1}{2}}$ and the problem becomes

$$\text{maximize} \quad g_{\text{Lap}}^T D^{-\frac{1}{2}} \{QD^{-\frac{1}{2}}g_{\text{Lap}} \}.$$

The eigenvalues of $D^{-\frac{1}{2}} Q D^{-\frac{1}{2}}$ are normalized and are between $-1$ and $1$ [47]. The greatest merit of this technique is that it normalizes the eigenvalues so that they are all standardized to be in the range between $-1$ and $1$. Despite this property, it is obvious that this normalization also satisfies the constraint of Equation (28). This is because $D - P$ is a non-negative definite matrix (the proof can be found in Appendix MP). Here, the objective function value is $1^T D1 = ||P||_1$. To justify the tighter solution bound property for the Laplacian normalization, we need to prove that $n \lambda_{\text{max}} P \geq ||P||_1$. In mathematics, the $l_1$ norm of the $n \times n$ matrix $P$ can be expressed as [48]

$$||P||_1 = \left(1 - \frac{c_x}{2}\right) n ||P||_2, \quad \text{where} \quad c_x = \sum_{i,j} \left(\frac{|P_{ij}|}{n ||P||_2} - \frac{1}{n}\right)^2. \quad (30)$$

With this equality, the justification becomes

$$\lambda_{\text{max}} P \geq \left(1 - \frac{c_x}{2}\right) ||P||_2. \quad (31)$$

In this equation, we can see that the inequality must hold if $c_x = 2$. This is because the right-hand-side becomes zero. On the other hand, if $c_x = 0$, the inequality is hard to be satisfied. This is because $||P||_2 = \sqrt{\sum \lambda_i^2}$, $\lambda_i$ is an eigenvalue of $P$. Because of this, we consider the following two different scenarios that appeared widely in many BQP problems. The first one is that the matrix $P$ is sparse and many of its elements are zero. The second one is that most elements are non-negative.

**Scenario 1:** the matrix $P$ is sparse. In many BQP applications such as signal restoration, many elements of the matrix $P$ are zero. The number of non-zero elements of the matrix $P$ is about $3n$ or $4n$. Because of this, we assume $\sum_{i,j} |P_{ij}|$ to be much less than $n$. We assume that $n \gg k(m_d)^2$, $m_d$ is the number of distinct values among $|P_{ij}|$. The distinct values are $q_1, q_2, \ldots, q_{m_d}$. Among $k$ non-zero elements, there are $k n$ whose values are $q_j$ for $j = 1, \ldots, m_d$. This implies that $\sum_{i,j} |P_{ij}| = k$. With these settings, we can compute $||P||_2$ as $\sqrt{n \sum_{j} q_j^2}$. Also, we have

$$c_x = 2 \left(1 - \frac{k \max_{j} q_j^2}{n} \sum_{j=1}^{m_d} \frac{k_j q_j}{k \max_{j} q_j} \right) \quad (32)$$

Here, $k \max_{j} q_j = \max_{j} |q_j|$, $k_j q_j$ is the jth entry of $k \max_{j} q_j$. The detailed derivation for Equation (32) is given in Appendix MP. Obviously, $\sqrt{\frac{k \max_{j} q_j}{n} \sum_{j=1}^{m_d} \frac{k_j q_j}{k \max_{j} q_j}} \leq 1$ and $\sum_{j=1}^{m_d} \frac{k_j q_j}{k \max_{j} q_j} \leq m_d$. The above expression becomes

$$c_x \geq 2 \left(1 - \frac{k \max_{j} q_j}{n} m_d \right) \quad (33)$$

Since $n \gg k(m_d)^2 \geq k \max_{j} q_j^2$, the inequality $\sum_{j=1}^{m_d} \frac{k_j q_j}{k \max_{j} q_j} \leq m_d \rightarrow 0$ and $c_x \approx 2$ and thus the inequality of Equation (31) is satisfied. The inequality in Equation (33) is verified in Appendix II.C.

**Scenario 2:** Most elements of the matrix $P$ are non-negative. BQP problems are also widely applied for data partitioning problems. The matrix $P$ is usually set as the distance matrix, which is non-negative. To simulate this scenario, we assume that $P_{ij}$ follows a uniform distribution with range $[0,1]$. Here, we have to examine if $n \lambda_{\text{max}} P \geq ||P||_1$ still holds.

(i) Closed form expression of $n \lambda_{\text{max}} P$: Since $P$ is a symmetric $n \times n$ random matrix with diagonal elements being zero, we can obtain the expected value of $\lambda_{\text{max}} P$. It is noted that $E(\lambda_{\text{max}} P) = \max_{v ||v||_2=1} v^T E(P) v$. The expected value of a uniform distribution with range $[0,1]$ is $\frac{1}{2}$. This leads to $E(P_{ij}) = \frac{1}{2}$ if $i \neq j$ and $E(P_{ij}) = 0$ if $i = j$. For this matrix, the eigenvector is $1/\sqrt{n}$ and thus, $E(\lambda_{\text{max}} P) = \frac{1}{n} 1^T E(P) 1 = \frac{n-1}{2}$. This gives $E(n \lambda_{\text{max}} P) = \frac{n(n-1)}{2}$.

(ii) Closed form expression of $||P||_1$: Since $P$ is a symmetric $n \times n$ random matrix with diagonal elements being zero, the upper triangular matrix is the same as the lower triangular matrix. In other words, only half elements of the matrix are random. Because of this, we have $E(||P||_1) = \sum_{ij} E(P_{ij}) = \frac{1}{2} \sum_{i \geq j} E(P_{ij}) = \frac{n(n-1)}{2}$. The last equality holds because the expected value of a uniform distribution with range $[0,1]$ is $\frac{1}{2}$.
Based on (i) and (ii), we can see that the inequality still holds, meaning that the binary solution obtained by Laplacian normalization is as tight as the spectral relaxation. By combining the analysis of two scenarios, we can conclude that the Laplacian normalization gives a tighter solution bound than that of the spectral relaxation.

\[
\text{Input: } A_{\text{ineq}}, A_{\text{eq}}, b_{\text{ineq}}, b_{\text{eq}}, Q \\
\text{Output: } v
\]

Step 1: Find the eigenvector \( v_{\text{max}} \) corresponding to the largest eigenvalue of \( Q \). Take \( s = \text{sign}(v_{\text{max}}) \).

Step 2: If it is a constrained BQP problem,

Step 2(i): Compute \( d = \max(b_{\text{ineq}} - A_{\text{ineq}}s, 0) \) and \( d_- = \max(b_{\text{ineq}} - A_{\text{ineq}}(-s), 0) \). If \( \|d\|_2 < \|d\|_2 \), \( v = d_-, s = -s \).

Step 2(ii): Compute \( \lambda_{\text{GBQP}} = \frac{\|Q\|_2/\|A_{\text{eq}}s - b_{\text{eq}}\|_2^2 + \|A_{\text{ineq}}s + d - b_{\text{ineq}}\|_2^2}{\|A_{\text{eq}}s - b_{\text{eq}}\|_2} \).

Step 2(iv): Compute \( p = \left[ -\lambda_{\text{GBQP}} \left( |b_{\text{eq}}| + |b_{\text{ineq}} - d^T\right|_2 \right] \lambda_{\text{GBQP}} \left( b_{\text{eq}}^T A_{\text{eq}} + (b_{\text{ineq}} - d^T) A_{\text{ineq}} \right)^T \left[ Q - \lambda_{\text{GBQP}} \left( A_{\text{eq}}^T A_{\text{eq}} + A_{\text{ineq}}^T A_{\text{ineq}} \right) \right] \).

Step 2(vi): Find the eigenvector \( v = \frac{x_0}{\|x_0\|} \) corresponding to the largest eigenvalue of \( P_{\text{nor}} \).

Step 2(vii): Output \( v = x_0 \).

Step 3: If it is an unconstrained BQP problem,

Step 3(i): Compute \( D_{\text{il}} = \sum_{j} P_{ij} \) and \( P_{\text{nor}} = D^{-1/2} P D^{-1/2} \).

Step 3(ii): Find the eigenvector \( x \) corresponding to the largest eigenvalue of \( P_{\text{nor}} \).

Step 3(iii): Output \( v = x \).

ALGO 6. Initialization of the Proposed Solver.

The complete initialization algorithm is given in ALGO 6. It basically contains two parts. One is for the constrained problem (Step 2) while the other one is the unconstrained problem (Step 3). For the constrained problem, if the equality or inequality constraints are missing, the corresponding coefficients are taken as zeros. This is to delete their contributions in the calculation. After that, we compute the vectors \( d, \lambda_{\text{GBQP}}, P, P_{\text{nor}} \) according to the equations shown above. For the inequality problem, we further require that the vector \( s \) should be close to the boundary of the feasible region. In Step 2(ii), we consider the negation of the vector \( s \) and compare the \( L_2 \) norms of \( d \) (without negation) and \( d_- \) (with negation) and pick the smaller one. The smaller \( L_2 \) norm value means that the vector \( s \) is closer to the boundary of the feasible region. After Step 2(vi), we output the initial guess, which is \( x \). It is noted that if \( x_0 = -1 \), the multiplication \( x_0 \left[ \begin{array}{c} x_0 \\ x \end{array} \right] \) will turn it to \( \left[ \begin{array}{c} 1 \\ x_0 x \end{array} \right] \) and thus the first entry must be one. For the unconstrained case, it does not have both equality and inequality constraints. We can directly apply the techniques introduced in Section 3.3.3 and relax the binary constraint and estimate by the spherical constraint.

**Discussions on the Proposed Solver**

We discuss the proposed method in the following aspects:

(a) Computation Complexity of the Proposed Method

The proposed method has two computationally expensive steps: finding the leading eigenvector of a matrix and the matrix-vector multiplication. It is well known that the worst case complexity of finding the eigenvector is \( O(n^3) \) whereas the matrix-vector multiplication is also \( O(n^2) \). Therefore, the computational complexity of the proposed solver is \( O(n^3) \), which is much less than the state-of-the-art methods. To our knowledge, the state-of-the-art SDCut method currently has the lowest computational complexity [33] but the complexity of each step of SDCut is \( O(kn^2) \) [33], where \( k \) is the number of times the eigenvalue decomposition is applied.

(b) Extension to More General BQP Problems

The proposed solver can be extended to solve more general BQP problems as below

\[
\max_{s \in \{-1,1\}^n} s^T Q s + 2 b^T s \quad s.t. \quad s \in \Omega.
\]

(34)

This can actually be written as the stated BQP problem

\[
\max_{s \in \{-1,1\}^n} s^T Q_b s_b \quad \text{with } Q_b = \left[ \begin{array}{c} 1 \\ b^T \\ Q \end{array} \right] \quad \text{and } s_b = \left[ \begin{array}{c} s^T \\ 0 \end{array} \right].
\]

Obviously, \( s^T Q_b s_b = s^T Q s + 2 b^T s + s_b^2 \) and \( s_b^2 = 1 \). We can then solve \( s^T Q_b s_b \) using the proposed solver. The optimal solution \( s^* \) can then be obtained by \( s^* = s_b \).

**4 EXPERIMENTAL RESULTS**

We verify the robustness of the proposed method (The source code is available at [https://github.com/benson-sylam/BQP]) and compare its performance with four other methods for the U.BQP and C.BQP problems: the spectral method, SeDuMi [49], SDPT3 [50], and SDCut [51], and the Exact Penalty Method (EPM) [41] for partition constraint problem as this method is implemented for this type of problem only. Each method is tested on Matlab R2010b 64-bit under Windows 10 (64-bit), Intel® Core™ i5-3450 CPU@ 3.10 GHz. RAM 8.00GB. The computer source codes of SeDuMi [49], SDPT3 [31,32,50], SDCut [51] and EPM for partition constraint [52] are publicly available. We use their default settings. The use of SDCut requires the L-BFGS-B package [53] (with mex files) to optimize the computation. However, the suggested L-BFGS-B package is not useable.
This influences the computation time of SDCut. Therefore, we only focus on the accuracy of this method in our comparison. For the eigenvalue decomposition, we use the built-in Matlab function.

We now discuss the use of spectral and SDR methods to obtain binary solutions that satisfy all constraints. The common approach by which the SDR methods (SeDuMi, SDPT3, and SDCut) obtain the binary vector is called random hyperplane rounding applied to the positive semidefinite matrix $S$ [30]. The computer code for this method is also downloaded from the web [51] and we use the default parameter setting. The idea of random hyperplane rounding is to generate a set of score vectors based on a multivariate normal distribution with zero mean and covariance matrix $S$. The binary vector is then obtained by taking the sign function of the score vector. This is the case for U.BQP problem. However, for the C.BQP problem, a simple sign function or thresholding cannot be used to obtain a binary vector that satisfies all constraints. Since the SDR methods are designed for linear equality constraints and quadratic constraints problems, in order to handle linear inequality constraint problems, we decompose the inequality constraints into their corresponding equality constraints and only consider the solutions along the boundary. Concretely, the equivalent equality-constrained problem is solved to obtain $p$ covariance matrices, where $p$ is the total number of constraints. The random hyperplane rounding technique is applied to each of the covariance matrices to get $N$ score vectors. Each of the score vectors are then projected to the constraint by optimizing the following problem:

$$\max_{s \in \{-1,1\}^N} s^T q_{\text{score}} \text{ s.t. } s^T a = b$$  (35)

where $q_{\text{score}}$ is the score vector. The vectors $a$ and $b$ are coefficient and the right-hand-side values of the $p$-constraints. By introducing the Lagrange multiplier $L_{\text{score}}$, the dual problem is

$$\min_{L_{\text{score}}} \max_{s \in \{-1,1\}^N} s^T (q_{\text{score}} + L_{\text{score}} a) + L_{\text{score}} b$$

$$= \min_{L_{\text{score}}} \Vert q_{\text{score}} + L_{\text{score}} a \Vert_1 + L_{\text{score}} b.$$  (36)

This is again a convex optimization problem and is similar to Equation (11). Thus, it can be solved by ALGO 2 (Find_StepSize). This procedure is applied to each of constraints. That means, the above procedure is repeated $p$ times. The one that satisfies all the constraints and maximizes the objective function is the optimal solution for the problem. For the spectral method, the binary vector is taken as a sign function of the leading eigenvector of the matrix $Q$ for the U.BQP problem. For the C.BQP problem, we treat the leading eigenvector as the score vector and apply the above procedure to each constraint and obtain the optimal solution. In each experiment below, all the solutions satisfy all of the problem’s constraints.

### 4.1 Tests on Random Matrices

We test the proposed solver with random matrices. The construction of matrix $Q$ as follows. Each element of the upper triangular part of matrix $Q$ is drawn from the uniform distribution with range $[-10,10]$. The lower triangular part of matrix $Q$ is then obtained by transposing the upper triangular part of the matrix. The procedure is repeated 30 times. The average performances using different solvers are shown in Table 1. In this experiment, we also compare the performance of $l_p$-Box method [54], which relaxes the binary constraint to $l_p$ box constraints. We downloaded the Matlab code from the web [55].

<table>
<thead>
<tr>
<th>n</th>
<th>1000</th>
<th>300</th>
<th>500</th>
<th>700</th>
<th>10^3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Spectral</td>
<td>SDCut</td>
<td>SeDuMi</td>
<td>SDPT3</td>
<td>$l_p$-Box</td>
</tr>
<tr>
<td>Obj.</td>
<td>1101.28</td>
<td>1270.8</td>
<td>1270.9</td>
<td>1270.3</td>
<td>1101.2</td>
</tr>
<tr>
<td>Time</td>
<td>0.00</td>
<td>5.30</td>
<td>4.21</td>
<td>4.05</td>
<td>0.31</td>
</tr>
<tr>
<td>Obj.</td>
<td>5658.6</td>
<td>6333.8</td>
<td>6340.2</td>
<td>6340.3</td>
<td>5655.9</td>
</tr>
<tr>
<td>Time</td>
<td>0.01</td>
<td>29.23</td>
<td>11.86</td>
<td>9.13</td>
<td>0.11</td>
</tr>
<tr>
<td>Obj.</td>
<td>12050.9</td>
<td>13279.1</td>
<td>13288.6</td>
<td>13285.3</td>
<td>12044.6</td>
</tr>
<tr>
<td>Time</td>
<td>0.01</td>
<td>95.32</td>
<td>32.02</td>
<td>19.51</td>
<td>0.21</td>
</tr>
<tr>
<td>Obj.</td>
<td>19868.0</td>
<td>21723.2</td>
<td>21733.6</td>
<td>21788.5</td>
<td>19872.8</td>
</tr>
<tr>
<td>Time</td>
<td>0.02</td>
<td>215.40</td>
<td>63.01</td>
<td>33.46</td>
<td>0.55</td>
</tr>
<tr>
<td>Obj.</td>
<td>33831.5</td>
<td>36526.3</td>
<td>36504.1</td>
<td>36522.7</td>
<td>37909.2</td>
</tr>
<tr>
<td>Time</td>
<td>0.05</td>
<td>596.47</td>
<td>163.62</td>
<td>78.81</td>
<td>1.38</td>
</tr>
</tbody>
</table>

The leftmost column shows the size $n$ of an $n \times n$ matrix $Q$. The second column shows the objective function value, i.e., $s^T Q s$ (Obj.), and the computational time (Time in second). Because the U.BQP problem is a maximization problem, a larger objective function value implies a better result. In the table, the proposed solver not only gives the best objective function value but also obtains solutions within a few seconds and is up to 150 times faster than SeDuMi and SDPT3. Although the spectral method and the $l_p$-Box method are fast, their performances are not as good as the proposed method. Moreover, the difference in accuracy between the proposed solver and all other solvers, are larger when the sample size $n$ is larger. Theoretical, SDR methods provide a tighter solution bound and give higher solution qualities. However, SDR methods are generally computational expensive. They need to apply the random hyperplane rounding technique to obtain a binary solution. The technique is an independent random search and the binary vector generated each time is independent of the previous one. Therefore, a new binary vector is not necessarily better than the previous one. Even if the global optimal solution is next to a trial vector, the newly generated vector can be farther away from it. This lengthens the searching process and a large number of trials is needed in order to get a high-quality solution. The problem becomes more severe if the dimension of the vector space is higher. In contrast, the proposed solver examines the stochastic neighborhood of a solution and search for a local optimal
solution therein. This introduces a more efficient way to find an optimal solution. If the high-quality solution is next to a trial solution, the proposed solver can swiftly drive the trial solution to the stochastic neighborhood of the optimal solution and obtain a better result. This saves much computation time.

We also perform similar random tests for the equality and inequality constraint problems. Moreover, we also tested the proposed solver with different sets of random matrices. The results are excellent and the details are shown in Appendix I.

![Fig. 3. Signal restoration using different solvers with $c_{\text{noise}} = 0.5$, $n = 1000$, and $\mu = 1$.](image)

4.2 Binary Restoration

A corrupted binary signal can be recovered by minimizing the following objective function [25,26]

$$
\min_{s_i \in \{-1,1\}_n} \sum_i (s_i - \hat{s}_i)^2 + \mu \sum_{i \in N(i)} \sum_j (s_i - s_j)^2,
$$

(37)

where $\{\hat{s}_j\}_{i=1}^n$ is the corrupted signal, $\mu$ is a user-defined parameter that facilitates continuity across the spatial neighborhood, and $N(i)$ indicates a spatial neighborhood of $i$. In binary restoration, the signal $\{\hat{s}_i\}, i = 1, \ldots, n$ is assumed to take the value $\pm 1$. It is noted that Equation (37) formulates a U.BQP problem in the form of Equation (34). That is,

$$
\sum_i (s_i - \hat{s}_i)^2 + \mu \sum_{i \in N(i)} (s_i - s_j)^2
= \sum_i (1 - 2s_i \hat{s}_i + \hat{s}_i^2) + \sum_{i \in N(i)} (2 - 2s_i s_j)
= c_{\text{values}} + 2 \left( - \sum_i s_i \hat{s}_i + \sum_{i \in N(i)} s_i s_j \right),
$$

(38)

where $c_{\text{values}}$ are the sum of the terms that do not involve $s_i$. The objective of this problem is to restore the signal from the corrupted signal by solving the optimization problem in Equation (37). The binary solution vector $\mathbf{s}$ is the cleansed version of the input signal. The matrix $Q$ for this binary model consists of many zero elements. Thus, less computation and storage are required for our testing platform. We test the robustness of different solvers in (i) a one-dimensional case and (ii) a two-dimensional case (i.e., image).

(i) One-Dimensional Case

A one-dimensional signal $\{\hat{s}_i\}_{i=1}^n$ is shown in Figure 3(a). The signal is corrupted by normally distributed noise with standard derivation $c_{\text{noise}}$. The corrupted signal with $c_{\text{noise}} = 0.5$ are shown in Figure 3(b). For binary restoration, the neighborhood $N(i)$ is taken as $\{i-1, i, i+1\}$. The binary solution vector $\mathbf{s}$ is the cleansed version of the input noisy signal. The performances of different solvers are shown in TABLE 2. In addition to the objective function value (Obj.) and computational time (Time), we also compute the error measure (Error). It is computed as the $l_1$ norm difference between the recovered signal and the original signal. The proposed solver performs the same as the SDR methods, i.e., SDCut, SeDuMi, and SDPT3, but obtains the solutions much more quickly. Figure 3(d) shows the recovered signal. We also compare the proposed solver and the other solvers using different parameter settings. This can be found in Appendix II(A). Besides, we apply the proposed solver with different initial guesses, i.e. the eigenvectors with the second largest eigenvalues. We obtain the same objective function values but with a longer computation time. This shows that the proposed solver is robust to initial guess.

![Table 2. One-dimensional signal restoration using different solvers](image)

(ii) Two-Dimensional Case

The binary restoration model is applied to a set of binary images with different sizes. They are corrupted by “salt and pepper” noise at 30%, 50%, and 70% levels. For large images, SDCut, SeDuMi, and SDPT3 run out of memory. In these cases, we need only compare the performances between the spectral method and the proposed method. The spatial neighborhood $N(i,j)$ of the binary restoration model is taken as $\{(i-1,k), (i+1,k), (i,k-1), (i,k+1)\}$. The binary solution vector $\mathbf{s}$ is the cleansed version of the input noisy image. The experiments in this section are divided into two parts according to the size of the images. Given the size $n_1 \times n_2$ of a binary image, the two parts are (a) small images ($n_1, n_2 \leq 100$) and (b) large
images ($200 \leq n_1, n_2 \leq 500$).

(a) Small Images

Figure 4 shows four images together with their corrupted versions. The performances of the proposed solver and all SDR methods are very similar. The qualitative results of the proposed solver can be found in Figure 4. The results of other methods can be found in Appendix II(B). However, the computational time of the proposed method is much shorter than that of SeDuMi and SDPT3, as shown in TABLE 3. In all cases, the proposed solver takes under a second to obtain the result.

Table 3. Computational time of different solvers (time in sec).

<table>
<thead>
<tr>
<th></th>
<th>Spectral</th>
<th>SeDuMi</th>
<th>SDPT3</th>
<th>Proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Banana 30%</td>
<td>0.004297</td>
<td>288.764</td>
<td>38.2935</td>
<td>0.023142</td>
</tr>
<tr>
<td>50%</td>
<td>0.004106</td>
<td>270.7541</td>
<td>39.33172</td>
<td>0.031951</td>
</tr>
<tr>
<td>Dog 30%</td>
<td>0.005555</td>
<td>1525.332</td>
<td>106.2322</td>
<td>0.048618</td>
</tr>
<tr>
<td>50%</td>
<td>0.005181</td>
<td>1752.792</td>
<td>106.0853</td>
<td>0.050794</td>
</tr>
<tr>
<td>Man &amp; Woman 30%</td>
<td>0.008985</td>
<td>8091.136</td>
<td>163.3371</td>
<td>0.051355</td>
</tr>
<tr>
<td>50%</td>
<td>0.286249</td>
<td>8367.086</td>
<td>353.6177</td>
<td>0.051137</td>
</tr>
<tr>
<td>Square 30%</td>
<td>0.005646</td>
<td>3051.247</td>
<td>154.1549</td>
<td>0.045029</td>
</tr>
<tr>
<td>50%</td>
<td>0.005378</td>
<td>2295.831</td>
<td>127.8136</td>
<td>0.046731</td>
</tr>
</tbody>
</table>

Fig. 4. Original and corrupted images with 30% and 50% noise, and the restored images using the proposed solver with $\mu = 0.05$.

(b) Large Images

Seven large images are used in this set of experiments. Each image is corrupted with 50% and 70% “salt and pepper” noise. Each image is tested with three parameter settings, $\mu = 0.5$, $\mu = 1$, and $\mu = 2$. Because the matrices are too large, the SDR methods, i.e., SDCut, SeDuMi, and SDPT3, run out of memory. In these cases, we need only compare the performances between the spectral method and the proposed method. Figure 5 show the qualitative results of the two BQP solvers. The quantitative results can be found in Appendix II(B). Obviously, the proposed method gives much better results than the spectral method. For the rice image, the proposed method gives a nearly black image when the value of $\mu$ is large because the regularization parameter is too large for this image and most of the recovered values are regularized to be the same zero value. The proposed method therefore accurately reflects the regularization property. On the other hand, the spectral method fails and its recovered images resemble the noisy images. We also compare the performance of different solvers by inverting the colors of the images. That is, turning the black to white and white to black. The proposed solver still performs better than that of other solvers. The results can be found in Appendix II(C).

The superior performance of the proposed solver over the spectral method is attributed to the fact that the solution obtained by the proposed solver satisfies the KKT optimality conditions. This is proved in Theorem 3. The KKT optimality conditions guarantee that the obtained solution is a local optimal solution. This means it is better than its stochastic neighborhoods. Moreover, with the stochastic neighborhood search, the proposed solution is guaranteed to be better than many of its stochastic neighborhoods. However, for the spectral method, no optimality conditions are guaranteed. The spectral solution may not be as good as its neighborhoods. Although the spectral method is faster than that of the proposed method, the computation of the proposed method mainly involves matrix-vector multiplication and the computation time is still low.

4.3 Bisection Problems with Equally Sized Partitions

We apply the solvers to the graph bisection problems with partitions of equal size in two different ways: (i) data bisection and (ii) image bisection. A bisection of a graph is to partition a set of vertices into two groups in which the number of vertices in each part differs by no more than 1. In image processing and pattern recognition applications [1], this is done to minimize the total weights of edges. This method can be formulated as $\min_{s \in \{-1,1\}^n} s^T W s \text{ s.t. } s^T 1 = 0$, where $W = \Lambda - Q$ is the graph Laplacian matrix. $Q \in \mathbb{R}^{n \times n}$ is the weighted affinity matrix, $\Lambda$ is the degree matrix. The binary solution vector $s$ is the indicator function. The element of the vectors equals to 1 if it belongs to one cluster. The element is -1 if it belongs to another cluster. Because $s^T \Lambda s$ is a constant for $s \in \{-1,1\}^n$ and $s^T W s = s^T \Lambda s - s^T Q s$, the above problem becomes $\max_{s \in \{-1,1\}^n} s^T Q s \text{ s.t. } s^T 1 = 0$. The similarity matrix $Q$ is calculated based on the weighted distance between points $i$ and $j$, as $Q_{ij} = \exp \left(-\frac{d(i,j)^2}{\sigma^2}\right)$, where $\sigma$ is a user defined parameter.

A. Data Bisection

The vertices and edges for this problem are the data points and the distances among the data, respectively. The distance function $d(i,j)$ is defined as the Euclidean distance between data points $x_i$ and $x_j$. In this experiment, we also...
compare the performance of EPM.

The construction of two clusters of data with noise corruption is as follows. The two clusters are randomly drawn from two normal distributions $N(0, I)$ and $N(10I, I)$. The size of the two clusters are $\text{ratio} \times n$ and $(1 - \text{ratio}) \times n$, where ratio is a value between 0 and 1. After that, $0.2n$ noisy data points from the normal distribution $N(10I, 10I)$ are added to the dataset. Therefore, the size of the dataset is $1.2 \times n$.

<table>
<thead>
<tr>
<th>Images</th>
<th>Spectral</th>
<th>Proposed</th>
<th>Spectral</th>
<th>Proposed</th>
<th>Spectral</th>
<th>Proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\mu = 0.5$</td>
<td>$\mu = 1$</td>
<td>$\mu = 2$</td>
<td>$\mu = 2$</td>
<td>$\mu = 2$</td>
<td>$\mu = 2$</td>
</tr>
<tr>
<td>Horse, Size 231 × 250</td>
<td>![Horse Image]</td>
<td>![Horse Image]</td>
<td>![Horse Image]</td>
<td>![Horse Image]</td>
<td>![Horse Image]</td>
<td>![Horse Image]</td>
</tr>
<tr>
<td>50%</td>
<td>![Horse Image]</td>
<td>![Horse Image]</td>
<td>![Horse Image]</td>
<td>![Horse Image]</td>
<td>![Horse Image]</td>
<td>![Horse Image]</td>
</tr>
<tr>
<td>70%</td>
<td>![Horse Image]</td>
<td>![Horse Image]</td>
<td>![Horse Image]</td>
<td>![Horse Image]</td>
<td>![Horse Image]</td>
<td>![Horse Image]</td>
</tr>
<tr>
<td>Fingerprint, Size 600 × 748</td>
<td>![Fingerprint Image]</td>
<td>![Fingerprint Image]</td>
<td>![Fingerprint Image]</td>
<td>![Fingerprint Image]</td>
<td>![Fingerprint Image]</td>
<td>![Fingerprint Image]</td>
</tr>
<tr>
<td>50%</td>
<td>![Fingerprint Image]</td>
<td>![Fingerprint Image]</td>
<td>![Fingerprint Image]</td>
<td>![Fingerprint Image]</td>
<td>![Fingerprint Image]</td>
<td>![Fingerprint Image]</td>
</tr>
<tr>
<td>70%</td>
<td>![Fingerprint Image]</td>
<td>![Fingerprint Image]</td>
<td>![Fingerprint Image]</td>
<td>![Fingerprint Image]</td>
<td>![Fingerprint Image]</td>
<td>![Fingerprint Image]</td>
</tr>
</tbody>
</table>

Fig. 5. Comparison of spectral and proposed solvers. First column: corrupted images. Second to last columns: results of spectral and proposed methods.

TABLE 4. Data bisection using different solvers.

<table>
<thead>
<tr>
<th>n</th>
<th>$\sigma$</th>
<th>Spectral</th>
<th>SDCut</th>
<th>SeDuMi</th>
<th>SDPT3</th>
<th>EPM</th>
<th>Proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>0.5</td>
<td>37649.92</td>
<td>37578.49</td>
<td>37588.37</td>
<td>37582.02</td>
<td>37775.31</td>
<td>37948.91</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Obj.</td>
<td>Time</td>
<td>Obj.</td>
<td>Time</td>
<td>Obj.</td>
<td>Time</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>0.056584</td>
<td>5543.769</td>
<td>5543.769</td>
<td>563.2585</td>
<td>183.8989</td>
<td>2.515999</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>99427.9</td>
<td>98896.57</td>
<td>98896.57</td>
<td>98896.57</td>
<td>99086.24</td>
<td>100139.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Obj.</td>
<td>Time</td>
<td>Obj.</td>
<td>Time</td>
<td>Obj.</td>
<td>Time</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>0.034889</td>
<td>1383.754</td>
<td>1383.754</td>
<td>391.4917</td>
<td>133.4725</td>
<td>2.232679</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>200891.6</td>
<td>203176.4</td>
<td>203176.4</td>
<td>203198.9</td>
<td>200891.6</td>
<td>203367.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Obj.</td>
<td>Time</td>
<td>Obj.</td>
<td>Time</td>
<td>Obj.</td>
<td>Time</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>0.043833</td>
<td>706.9716</td>
<td>706.9716</td>
<td>663.219</td>
<td>201.2234</td>
<td>2.363029</td>
</tr>
</tbody>
</table>
Table 4 shows the results for \( n = 1000 \). Again, the proposed method shows the best performance in terms of objective function value and requires about 2 to 5 seconds to obtain the solutions. Now, we discuss the merits of the proposed solver over EPM. The proposed solver finds an optimal solution in the stochastic neighborhood constructed by the random-constrained bootstrapping technique. These stochastic neighborhoods are not restricted to a local convex ball. On the other hand, the EPM relaxes the original BQP problem to a biconvex problem and only requires the solution to be optimal in a local convex ball. If the initial guess is far away from the good solution, it may be trapped to a local optimal solution. This also explains why the EPM performs better than the spectral method only in some experiments. The EPM can perform worse than the spectral method if \( \sigma \) is larger. For a large \( \sigma \), the distances between two nodes are regularized and they are similar to each other. In other words, different partitions may produce similar results. This introduces many local optimal solutions and hinders the performance of EPM. However, this is not the case for the proposed solver. The proposed random constraint can effectively broaden the stochastic neighborhood ranges. This results in a higher chance of obtaining a good solution. In Appendix III(A), we also compare the performances of the proposed solver with different methods under different settings including the sample size \( n \) and the parameter \( \sigma \). The results of the proposed solver are excellent under different settings.

### B. Image Bisection

In this experiment, we apply the bisection model to real-world images. The distance \( d(i,j)^2 \) is taken as \( d(i,j)^2 = ||\text{Im}(i) - \text{Im}(j)||^2 + ||i - j||^2 \), where \( i \) and \( j \) are indices of the image. \( \text{Im}(i) \) is the \( i \)-th pixel value, and the similarity matrix \( Q \) is given by \( Q_{ij} = \exp\left(-\frac{d(i,j)^2}{\sigma^2}\right) \) with the parameter \( \sigma = 8/255^2 \). Unlike the binary restoration model, most of the elements of \( Q \) are not zero. Sparse computation and storage cannot be used. Because SeDuMi and SDCut have long computation times, we compare only the performances of the spectral method, SDPT3, EPM, and the proposed method. Four real-world images are used in this set of experiments. They are shown in Figure 6.

![Fig. 6. Test images for image bisection applications. From left to right: Dog, Flower 1, Flower 2, and Baby.](image)

In Table 5, the results show that the proposed solver performs the best in terms of the objective function value. Moreover, it is much faster than the SDR method. We also repeatedly apply the proposed solver with different random seeds to the above images. The objective function values remain the same. The results for different parameter settings for \( \sigma \) can be found in Appendix III(B).

We extend our experiments to three large images. All the SDR methods run out of memory because the images are too large. We compare only the performance of the spectral method, EPM, and the proposed method. The qualitative results are shown in Figure 7; the proposed method produces better bisection results than the spectral method and EPM. For the bird image, the spectral method and proposed method group the bird and branches together, which have similar pixel intensities. However, the solution quality obtained by EPM is not as good as the proposed method. It groups the background and branches together. Moreover, the objective function value of the proposed method is the highest among the methods.

![Table 5. Image bisection results for medium size images using different solvers with \( \sigma = 8/255^2 \)](image)

<table>
<thead>
<tr>
<th></th>
<th>Spectral</th>
<th>SDPT3</th>
<th>EPM</th>
<th>Proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dog (64x71)</td>
<td>2419274</td>
<td>2727217</td>
<td>1203285</td>
<td>2789654</td>
</tr>
<tr>
<td>Time</td>
<td>0.75912</td>
<td>3327.61</td>
<td>8.0281</td>
<td>14.4036</td>
</tr>
<tr>
<td>Flower 1 (60x60)</td>
<td>2197096</td>
<td>2229493</td>
<td>839260</td>
<td>2240117</td>
</tr>
<tr>
<td>Time</td>
<td>0.50896</td>
<td>1858.724</td>
<td>5.1305</td>
<td>11.4477</td>
</tr>
<tr>
<td>Flower 2 (46x47)</td>
<td>1019168</td>
<td>1075439</td>
<td>951428</td>
<td>1080022</td>
</tr>
<tr>
<td>Time</td>
<td>0.142015</td>
<td>448.3789</td>
<td>2.0077</td>
<td>1.8747</td>
</tr>
<tr>
<td>Baby (53x68)</td>
<td>2922105</td>
<td>2922067</td>
<td>1098211</td>
<td>2922135</td>
</tr>
<tr>
<td>Time</td>
<td>0.362451</td>
<td>2022.376</td>
<td>5.1412</td>
<td>9.0367</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bird, size: 103 x 81</th>
<th>Spectral</th>
<th>EPM</th>
<th>Proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obj</td>
<td>7992686</td>
<td>4382760</td>
<td>8794890</td>
</tr>
<tr>
<td>Rose, size: 91 x 85</td>
<td>Spectral</td>
<td>EPM</td>
<td>Proposed</td>
</tr>
<tr>
<td>Obj</td>
<td>16639739</td>
<td>7701870</td>
<td>16736239</td>
</tr>
<tr>
<td>Mountain, size: 81 x 144</td>
<td>Spectral</td>
<td>EPM</td>
<td>Proposed</td>
</tr>
<tr>
<td>Obj</td>
<td>10534220</td>
<td>7823264</td>
<td>13604064</td>
</tr>
</tbody>
</table>

![Fig. 7. Image bisection results for large images using spectral, EPM and proposed solver.](image)

### 4.4 General Graph Bisection Problems

The experiments in this section consider the image bisection problem as described in the previous section with more general constraints. Instead of constraining the problem with equally sized partitions, we consider the following two inequality constraints:
These two constraints restrict the ratio between +1 and −1 to between 0.2 × n and 0.4 × n. This setting allows the model to choose the best ratio between +1 and −1 that can maximize the objective function. In contrast with the constraint of equally sized partitions, these two constraints allow the model to partition objects with more complicated shapes.

Table 6 shows the bisection results of the medium size images shown in Figure 6. Again, because the image is too large for some of the compared methods, we compare only the performances of the spectral method, SDPT3, and the proposed method. The results clearly show that the proposed method has the best performance. As SDPT3 is applicable only for quadratic constraints or linear equality constraints, the linear inequality constraints in this problem are decomposed into two linear equality constraints, i.e. 0.2 × n = s^T 1 and 0.4 × n = s^T 1. SDPT3 has to apply twice, one for each of the two linear constraints, and the best solution is outputted. It is noted that if the optimal solution is an interior point and not a boundary point, SDPT3 may not be able to find a good quality solution. The results for applying different parameter setting of σ can be found in Appendix IV.

**TABLE 6. Image bisection results for medium size images using different solvers with σ = 8/255^2**

<table>
<thead>
<tr>
<th></th>
<th>Spectral</th>
<th>SDPT3</th>
<th>Proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dog (64x71)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Obj.</td>
<td>2743859.2</td>
<td>2892180.2</td>
<td>2892390.9</td>
</tr>
<tr>
<td>Time</td>
<td>1.28886</td>
<td>39070.073</td>
<td>35.4652</td>
</tr>
<tr>
<td>Flower 1 (60x60)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Obj.</td>
<td>2328006.2388</td>
<td>2345368.3673</td>
<td>2430740.7315</td>
</tr>
<tr>
<td>Time</td>
<td>8.5242</td>
<td>18146.6942</td>
<td>11.4421</td>
</tr>
<tr>
<td>Flower 2 (46x47)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Obj.</td>
<td>1479892.4</td>
<td>1493789.3</td>
<td>1494323.3</td>
</tr>
<tr>
<td>Time</td>
<td>0.194642</td>
<td>4279.9216</td>
<td>4.9826</td>
</tr>
<tr>
<td>Baby (53x68)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Obj.</td>
<td>2721067.9</td>
<td>2462427.9</td>
<td>2852949.8</td>
</tr>
<tr>
<td>Time</td>
<td>0.76642</td>
<td>25749.1206</td>
<td>15.0468</td>
</tr>
</tbody>
</table>

The optimal solution obtained by the proposed solver for Flower 1 is an interior point, not a boundary point (Refer to Section 3.1). That is, it does not satisfy the two equality constraints, 0.2 × n = s^T 1 and 0.4 × n = s^T 1, but it satisfies the inequality constraint 0.2 × n < s^T 1 < 0.4 × n. On the other hand, SDPT3 is not designed for linear inequality constraints. We can only find an optimal solution at the boundary of the feasible region. If we remove the two constraints in the formulation, SDPT3 will be solving a U.BQP problem and its optimal solution must be a vector of ones as all elements of the matrix Q are non-negative. This shows that the proposed solver introduces a better way to handle inequality constraint problem.

We also perform experiments on some large images. Again, because the images are too large, the SDR methods run out of memory. Therefore, we compare only the performance of the spectral method and the proposed method. Figure 8 shows the results. The proposed method produces better bisection results than the spectral method.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we present a novel approach to search the optimal solution of a BQP problem over stochastic neighborhoods. Our approach consists of two main phases. A potential solution is first obtained by the deterministic algorithm. We applied the project gradient ascent method and proved that it converges to a solution that satisfies all the KKT optimality conditions. The stochastic algorithm is then applied to this potential solution to obtain a stochastic neighborhood vector. The deterministic algorithm is then applied again to this stochastic neighborhood vector. If a better solution is found, this will be treated as a new potential solution. This procedure is repeated until the solution is better than many of its stochastic neighborhood vectors. As our solver adopts a dependent search, it has much less computation time comparing to SDR methods. Compared to EPM, our solver searches for the optimal solution in a much larger neighborhood ranges instead of just in a convex ball, and this enables it to escape more easily from locally optimal solutions. We have applied the proposed solver to different image processing, pattern recognition, and computer vision applications that can be formulated as either U.BQP or C.BQP problems. Experimental results show that our solver gives much superior results. Specifically, it produces better quality solution than the state-of-the-art methods but with a much shorter computation time.

![Fig. 8. Image bisection results for large images using spectral and proposed methods.](image-url)
neighborhood vectors in the vicinity of the potential solution, whereas CAB computes the neighborhood using a confidence-bound distance measure. Another difference is that once the neighborhood is determined, our solver search for the solution within it using deterministic gradient ascent, whereas CAB search for item within it that provides the maximum payoff. The proposed method also shares a similar spirit with an advanced stochastic optimization, namely online methods for the quantification problem (OMQP) [57]. Both apply the primal-dual methods to solve an optimization problem. However, their theoretical analyses are different as they are analyzing two different types of problems. The proposed method solves non-concave binary optimization problems while the OMQP solves nested concave online supervised learning problems.

Several possible future works that can further speed up the proposed solver are suggested below. First, the matrix Q of the objective function is symmetric and most steps of the proposed solver involves matrix-vector multiplication. An efficient matrix-vector multiplication implementation can lighten the computation burden. Moreover, the operations of finding stochastic neighborhood vectors and comparing them with the obtained solution are independent and can be parallelized. This can further speed up the proposed solver. In the proposed solver, the number of neighborhood vectors (i.e. the parameter m in the stochastic step) is currently set as 50. However, in many experiments, a smaller value will still produce the same results. Therefore, investigating the optimal number of stochastic neighborhood vectors can potentially improve the speed of the proposed solver.

ACKNOWLEDGMENT

The work described in this paper was partially supported by the grants from Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. UGC/FDS14/E03/14 and UGC/FDS14/P04/17). We also want to give thanks to the great support from the Big Data Intelligence Centre of the Hang Seng University of Hong Kong. We are thankful to the anonymous reviewers for their valuable comments that greatly helped to improve the paper.

REFERENCES


[55] https://github.com/wubaoyuan/Lpbox-ADMM


Benson Shu-Yan Lam is currently an Assistant Professor in the Department of Mathematics and Statistics at the Hang Seng University of Hong Kong. He received both BSc and MPhil in Mathematical Science from the Department of Mathematics at Hong Kong Baptist University. Upon his completion of PhD degree at City University of Hong Kong, he joined Griffith University (Brisbane, Australia) as a postdoctoral researcher for two years. His research interest is in the fields of machine learning, pattern recognition, image processing and text mining.

Alan Wee-Chung Liew is currently an Associate Professor with the School of ICT, Griffith University, Australia. Prior to joining Griffith University in 2007, he was an Assistant Professor (2004-2007) at the Department of Computer Science and Engineering, Chinese University of Hong Kong, and a Senior Research Fellow (2001-2004) at the Department of Electronic Engineering, City University of Hong Kong. His research interest is in the fields of machine learning, pattern recognition, medical imaging, computer vision, and bioinformatics, and he has published extensively in these areas. He also served on the technical program committee of many international conferences, and as associate editor on several journals, including the IEEE Transactions on Fuzzy Systems.