A Penalized Quadratic Convex Reformulation Method for Random Quadratic Unconstrained Binary Optimization

Karthik Natarajan[*] Dongjian Shi[†] Kim-Chuan Toh[‡]

June 15, 2013

Abstract

The Quadratic Convex Reformulation (QCR) method is used to solve quadratic unconstrained binary optimization problems. In this method, the semidefinite relaxation is used to reformulate it to a convex binary quadratic program which is solved using mixed integer quadratic programming solvers. We extend this method to random quadratic unconstrained binary optimization problems. We develop a Penalized QCR method where the objective function in the semidefinite program is penalized with a separable term to account for the randomness in the objective.

1 Introduction

Consider the quadratic objective function:

$$q(\boldsymbol{x}; \boldsymbol{c}, \boldsymbol{Q}) = \boldsymbol{x}^T \boldsymbol{Q} \boldsymbol{x} + \boldsymbol{c}^T \boldsymbol{x}, \tag{1.1}$$

and the corresponding quadratic unconstrained binary optimization:

$$\text{(QUBO)} \quad \beta(\boldsymbol{c}, \boldsymbol{Q}) \;=\; \max_{\boldsymbol{x} \in \{0,1\}^N} q(\boldsymbol{x}; \boldsymbol{c}, \boldsymbol{Q}), \tag{1.2}$$

where $\boldsymbol{Q}$ is a $N \times N$ real symmetric matrix which is not necessarily negative semidefinite, and $\boldsymbol{c}$ is a vector in $\Re^N$. Quadratic unconstrained binary optimization (QUBO) has applications in a number of

[*]Engineering Systems and Design, Singapore University of Technology and Design, 20 Dover Drive, Singapore 138682. Email: natarajan_karthik@sutd.edu.sg

[†]Department of Mathematics, National University of Singapore, 10, Lower Kent Ridge Road, Singapore 119076. Email: shidongjian@nus.edu.sg

[‡]Department of Mathematics, National University of Singapore, 10, Lower Kent Ridge Road, Singapore 119076. Email: mattohkc@nus.edu.sg

diverse areas including computer-aided design (Boros and Hammer [14], Jünger et. al. [26]), solid-state physics (Barahona [4], Simone et. al. [39]), and machine scheduling (Alidaee et. al. [1]). Several graph problems, such as the max-cut and the maximum clique problems can be reformulated as QUBO problems. As a result, QUBO is known to be NP-hard (see Garey and Johnson [19]). A variety of heuristics and exact methods that run in non-polynomial time have been proposed to solve QUBO problems. When all the off-diagonal components of $\boldsymbol{Q}$ are nonnegative, QUBO is solvable in polynomial time (see Picard and Ratliff [34]). In this case, QUBO is equivalent to the following linear programming relaxation:

$$\begin{aligned} \max_{\boldsymbol{x},\boldsymbol{X}} \quad & \sum_{i=1}^{N}\sum_{j=1}^{N} Q_{ij}X_{ij} + \sum_{i=1}^{N} c_i x_i \\ \text{s.t.} \quad & X_{ij} \le x_i, \quad X_{ij} \le x_j, \qquad i,j \in [N], i \le j \\ & x_i \in [0,1], \quad X_{ij} \in [0,1], \qquad i,j \in [N], i \le j, \end{aligned}$$

where $[N] = \{1, \ldots, N\}$. Two other instances of QUBO that are solvable in polynomial time are when: (a) The graph defined by $\boldsymbol{Q}$ is series-parallel (Barahona [3]) and, (b) $\boldsymbol{Q}$ is positive semidefinite and of fixed rank (Allemand et. al. [2]). For an in-depth discussion on polynomial time solvable instances of quadratic binary optimization problems, the reader is referred to the paper of Duan et. al. [17]. For general $\boldsymbol{Q}$ matrices, branch and bound algorithms to solve QUBO problems were proposed by Carter [15] and Pardalos and Rodgers [33]. Beasley [5] developed two heuristic algorithms based on tabu search and simulated annealing while Glover, Kochenberger and Alidaee [21] developed an adaptive memory search heuristic to solve binary quadratic programs. Helmberg and Rendl [25] combined a semidefinite relaxation with a cutting plane technique, and applied it in a branch and bound setting. More recently, second order cone programming has been used to solve QUBO problems (see Kim and Kojima [27], Muramatsu and Suzuki [31], Ghaddar et. al. [20]). The work most closely related to this paper is the Quadratic Convex Reformulation (QCR) method proposed by Billionnet and Elloumi [8] that we discuss in detail next.

1.1 The Quadratic Convex Reformulation Method

One approach to solve the QUBO problem is with the Quadratic Convex Reformulation (QCR) method proposed by Billionnet and Elloumi [8]. Their method is inspired from the semidefinite programming relaxations for discrete optimization problems developed in Körner [29], Shor [38] and Poljak, Rendl and Wolkowicz [35] among others. The QCR method is based on transforming the QUBO problem to an equivalent convex QUBO problem and then solving it with off-the-shelf mixed integer quadratic programming solvers. For a binary variable $x_i \in \{0,1\}$, we have $x_i^2 = x_i$ and hence $\boldsymbol{x}^T \text{diag}(\boldsymbol{u})\boldsymbol{x} = \boldsymbol{u}^T\boldsymbol{x}$ for any $\boldsymbol{u} \in \Re^N$, where $\text{diag}(\boldsymbol{u})$ is the diagonal matrix obtained from the vector $\boldsymbol{u}$. A simple perturbation idea is to choose a vector $\boldsymbol{u} \in \Re^N$, such that $\boldsymbol{Q} - \text{diag}(\boldsymbol{u})$ is negative semidefinite. Define:

$$q_{\boldsymbol{u}}(\boldsymbol{x};\boldsymbol{c},\boldsymbol{Q}) = \boldsymbol{x}^T\left(\boldsymbol{Q} - \text{diag}(\boldsymbol{u})\right)\boldsymbol{x} + (\boldsymbol{c}+\boldsymbol{u})^T\boldsymbol{x}, \tag{1.3}$$

and the associated quadratic unconstrained binary maximization problem with a concave quadratic objective:

$$\beta(\boldsymbol{u};\boldsymbol{c},\boldsymbol{Q}) \quad = \quad \max_{\boldsymbol{x}\in\{0,1\}^N} q_{\boldsymbol{u}}(\boldsymbol{x};\boldsymbol{c},\boldsymbol{Q}). \tag{1.4}$$

Then, $\beta(\boldsymbol{c},\boldsymbol{Q}) = \beta(\boldsymbol{u};\boldsymbol{c},\boldsymbol{Q})$. Since the objective function in (1.4) is concave, it is possible to use off-the-shelf mixed integer quadratic programming solvers such as CPLEX to solve it. This gives an exact solution method to solve the QUBO problem where in the preprocessing step, a perturbation vector $\boldsymbol{u}$ is chosen such that $\boldsymbol{Q} - \text{diag}(\boldsymbol{u})$ is negative semidefinite and then in the solution step, the convex binary quadratic programming problem is solved.

The simplest possible choice of the perturbation vector $\boldsymbol{u}$ is to use:

$$\boldsymbol{u}_{\text{eig}} = \lambda_{\max}(\boldsymbol{Q})\boldsymbol{e}, \tag{1.5}$$

where $\lambda_{\max}(\boldsymbol{Q})$ is the largest eigenvalue of the matrix $\boldsymbol{Q}$ and $\boldsymbol{e}$ is an N dimensional vector with all entries equal to 1. Clearly, $\boldsymbol{Q} - \text{diag}(\boldsymbol{u}_{\text{eig}})$ is negative semidefinite and the function $q_{\boldsymbol{u}_{\text{eig}}}(\boldsymbol{x};\boldsymbol{c},\boldsymbol{Q})$ is concave with respect to the decision variables. Such an eigenvalue based preprocessing method was first proposed by Hammer and Rubin [24].

Billionnet and Elloumi [8] developed an alternate preprocessing phase where the "optimal" choice of the parameter vector $\boldsymbol{u}$ was found by solving a semidefinite program (SDP). Their approach is based on evaluating the upper bound $\bar{\beta}(\boldsymbol{u};\boldsymbol{c},\boldsymbol{Q})$ of the optimal value of the QUBO problem obtained by solving the convex relaxation of problem (1.4):

$$\bar{\beta}(\boldsymbol{u};\boldsymbol{c},\boldsymbol{Q}) = \max_{\boldsymbol{x}\in[0,1]^N} q_{\boldsymbol{u}}(\boldsymbol{x};\boldsymbol{c},\boldsymbol{Q}). \tag{1.6}$$

The "optimal" vector $\boldsymbol{u}_{\text{opt}}$ is chosen such that it minimizes the upper bound $\bar{\beta}(\boldsymbol{u};\boldsymbol{c},\boldsymbol{Q})$ subject to the constraint $\text{diag}(\boldsymbol{u}) - \boldsymbol{Q} \succeq 0$. Let

$$\boldsymbol{u}_{\text{opt}} = \text{argmin}\left\{\bar{\beta}(\boldsymbol{u};\boldsymbol{c},\boldsymbol{Q}) \mid \text{diag}(\boldsymbol{u}) - \boldsymbol{Q} \succeq 0\right\}. \tag{1.7}$$

Using standard duality arguments, it was shown in [8] that $\boldsymbol{u}_{\text{opt}}$ is the optimal $\boldsymbol{u}$ vector in the following SDP:

$$\begin{aligned}\bar{\beta}(\boldsymbol{u}_{\text{opt}};\boldsymbol{c},\boldsymbol{Q}) = \quad & \min_{r,\boldsymbol{u}} \quad r \\ & \text{s.t.} \quad \begin{bmatrix} r & -(\boldsymbol{c}+\boldsymbol{u})^T/2 \\ -(\boldsymbol{c}+\boldsymbol{u})/2 & \operatorname{diag}(\boldsymbol{u})-\boldsymbol{Q}\end{bmatrix} \succeq 0,\end{aligned} \tag{1.8}$$

where the positive semidefiniteness constraint is for a matrix of size $(N+1)\times(N+1)$. The semidefinite program (1.8) is the dual to the classic semidefinite programming relaxation of the QUBO problem:

$$\begin{aligned}\bar{\beta}(\boldsymbol{u}_{\text{opt}};\boldsymbol{c},\boldsymbol{Q}) = \quad & \max_{\boldsymbol{x},\boldsymbol{X}} \quad \sum_{i=1}^{N}\sum_{j=1}^{N} Q_{ij}X_{ij} + \sum_{i=1}^{N} c_i x_i \\ & \text{s.t.} \quad \begin{bmatrix} 1 & \boldsymbol{x}^T \\ \boldsymbol{x} & \boldsymbol{X}\end{bmatrix} \succeq 0 \\ & \qquad X_{ii} = x_i, \qquad\qquad i \in [N],\end{aligned} \tag{1.9}$$

where $\boldsymbol{u}_{\text{opt}}$ is the optimal dual variables to the equality constraints $X_{ii} = x_i,\ i \in [N]$. The SDP relaxation in (1.9) has been considered by several authors including Körner [29], Shor [38] and Poljak, Rendl and Wolkowicz [35] among others. Billionnet and Elloumi [8] proposed the use of this semidefinite program as a preprocessing phase to find the optimal perturbation vector before applying an exact branch and bound method to solve the QUBO problem based on solving convex relaxations. In the numerical experiments, they showed that the relative gap between the optimum value of the QUBO problem and the continuous relaxation $\bar{\beta}(\boldsymbol{u}_{\text{opt}};\boldsymbol{c},\boldsymbol{Q})$ is about half the relative gap between the optimum value and $\bar{\beta}(\boldsymbol{u}_{\text{eig}};\boldsymbol{c},\boldsymbol{Q})$. Solving the QUBO problem with the CPLEX solver is also faster using the SDP based preprocessing step as compared to the eigenvalue based preprocessing step. Subsequently, Billionnet et. al. ([12]) extended the QCR method to 0-1 quadratic programming problem with linear constraints and to more general mixed-integer programs in [11, 10, 9]. Galli and Letchford [18] extended this approach to mixed-integer quadratically constrained quadratic programs.

1.2 The Main Problem

In this paper, we extend the QCR method to solve parametric quadratic unconstrained binary optimization problems:

$$\max_{\boldsymbol{x}\in\{0,1\}^N} \left\{ q(\boldsymbol{x};\boldsymbol{c},\boldsymbol{Q}) := \boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{c}^T\boldsymbol{x} \right\}, \quad \forall \boldsymbol{c}\in\mathcal{C}, \tag{1.10}$$

where $\boldsymbol{Q}$ is a fixed $N\times N$ real symmetric matrix, and the parameter vector $\boldsymbol{c}$ varies in a set $\mathcal{C}$. Our main goal is to find a common preprocessing vector $\boldsymbol{u}$ that is valid across all $\boldsymbol{c}\in\mathcal{C}$ such that $\operatorname{diag}(\boldsymbol{u})-\boldsymbol{Q}\succeq 0$, and the preprocessing vector is "optimal" in some sense. To find such a preprocessing vector $\boldsymbol{u}$ for all $\boldsymbol{c} \in \mathcal{C}$, we assume that the parameter vector $\tilde{\boldsymbol{c}}$ is random with a probability distribution denoted by P. The expected optimal objective value for the QUBO problem (1.10) averaged over the possible realizations of $\tilde{\boldsymbol{c}}$ is expressed as:

$$\mathbb{E}_P\left[\beta(\tilde{\boldsymbol{c}}, \boldsymbol{Q})\right] = \int_{\mathcal{C}} \max_{\boldsymbol{x} \in \{0,1\}^N} q(\boldsymbol{x}; \boldsymbol{c}, \boldsymbol{Q}) dP(\boldsymbol{c}). \tag{1.11}$$

Evaluating $\mathbb{E}_P\left[\beta(\tilde{\boldsymbol{c}}, \boldsymbol{Q})\right]$ is clearly challenging since we need to solve a set of NP-hard QUBO problems for each realization of $\boldsymbol{c}$.

To facilitate the analysis, we assume that the probability distribution P for the random vector $\tilde{\boldsymbol{c}}$ is not completely specified. Rather, the joint distribution P lies in a Fréchet class of multivariate joint distributions that consists of all multivariate joint distributions with fixed marginal distributions P_i for each component $\tilde{c}_i$; for more details, see for example [16]. We denote the Fréchet class of distributions as $\mathbb{P}(P_1, \ldots, P_N)$. Distributions in the Fréchet class differ with respect to the dependency structures between the fixed marginal distributions. Since the probability distribution is incompletely specified, we focus on the extremal multivariate joint distribution of the random parameter vector $\tilde{\boldsymbol{c}}$ that maximizes the expected optimal objective value of the quadratic unconstrained binary optimization problem over all distributions in the Fréchet class. The problem of interest is defined as:

$$\begin{aligned} \beta^* &= \sup_{P \in \mathbb{P}(P_1, \ldots, P_N)} \mathbb{E}_P\left[\beta(\tilde{\boldsymbol{c}}, \boldsymbol{Q})\right] \\ &= \sup_{P \in \mathbb{P}(P_1, \ldots, P_N)} \int_{\mathcal{C}} \max_{\boldsymbol{x} \in \{0,1\}^N} q(\boldsymbol{x}; \boldsymbol{c}, \boldsymbol{Q}) dP(\boldsymbol{c}). \end{aligned} \tag{1.12}$$

The main contributions of the paper are summarized next:

1. In Section 2, we develop an equivalent but computationally implementable reformulation to find the tight upper bound β^* that exploits the structure of the Fréchet class of distributions. This reformulation is used in developing the Penalized QCR method.

2. In Section 3, we develop a SDP relaxation using the continuous relaxation of the reformulation to find a weaker upper bound on the optimal expected value β^*. The SDP relaxation has a natural interpretation as a Penalized Quadratic Convex Reformulation for QUBO problems with a random objective for the Fréchet class of distributions. Using this semidefinite program, we identify an "optimal" preprocessing vector $\boldsymbol{u}$ for this class of random QUBO problems.

3. In Section 4, we provide an extensive comparison between different approaches to solve QUBO problems with random objective coefficients. We demonstrate that for problems with up to 100

variables, the Penalized QCR method developed in this paper has computational advantages over alternate preprocessing approaches in terms of computational times and the quality of the bounds before concluding in Section 5.

2 A Tight Upper Bound on the Expected Optimal Value

In this section, we develop a reformulation for (1.12) to evaluate the tight upper bound on the expected optimal value of QUBO problem. Our approach is based on the results in Meilijson and Nadas [30] who developed a convex majorization approach to compute the tightest upper bound on the expected length of a critical path in a project network for the Fréchet class of distributions. Weiss [42] generalized this bound to linear combinatorial optimization problems such as the shortest path, maximum flow, and the reliability problem. Extensions of this approach to limited information on the marginal distributions have been proposed in Klein Haneveld [28], Birge and Maddox [13], Bertsimas, Natarajan and Teo [6, 7] and Natarajan, Song and Teo [32] among others. The main result in these papers is outlined in the next proposition.

Proposition 1. *(Meilijson and Nadas [30], Weiss [42]) Consider the linear combinatorial optimization problem:*

$$\max_{\boldsymbol{x} \in \mathcal{X} \subseteq \{0,1\}^N} \boldsymbol{c}^T \boldsymbol{x}.$$

For each $i \in [N]$, assume that the objective coefficient $\tilde{c}_i$ is a continuously distributed random variable with marginal distribution P_i. Define α^* to be the tight upper bound on the expected optimal objective value of the linear combinatorial optimization problem over the Fréchet class of distributions:

$$\alpha^* = \sup_{P \in \mathbb{P}(P_1,\ldots,P_N)} \mathbb{E}_P \left[\max_{\boldsymbol{x} \in \mathcal{X} \subseteq \{0,1\}^N} \tilde{\boldsymbol{c}}^T \boldsymbol{x} \right], \tag{2.1}$$

Then,

$$\alpha^* = \min_{\boldsymbol{d} \in \Re^N} \left(\max_{\boldsymbol{x} \in \mathcal{X} \subseteq \{0,1\}^N} \boldsymbol{d}^T \boldsymbol{x} + \sum_{i=1}^N \mathbb{E}_{P_i} [\tilde{c}_i - d_i]^+ \right). \tag{2.2}$$

For project networks, the reformulation in (2.2) can be interpreted as finding reference values $\boldsymbol{d}$ for the random durations of the activities, such that the project completion time based on $\boldsymbol{d}$ is balanced with the sum of the expected delays of the activity durations beyond $\boldsymbol{d}$. Using a similar approach, we develop a reformulation for the expected optimal objective value of QUBO problems for the Fréchet class of distributions in the next proposition.

Proposition 2. *For each $i \in [N]$, assume that the marginal distribution P_i of the continuously distributed random variable $\tilde{c}_i$ is given. Define*

$$\beta^* = \sup_{P \in \mathbb{P}(P_1,\ldots,P_N)} \mathbb{E}_P \left[\max_{\boldsymbol{x} \in \{0,1\}^N} \left(\boldsymbol{x}^T \boldsymbol{Q} \boldsymbol{x} + \tilde{\boldsymbol{c}}^T \boldsymbol{x} \right) \right], \tag{2.3}$$

and

$$\beta^{**} = \min_{\boldsymbol{d} \in \Re^N} \left(\max_{\boldsymbol{x} \in \{0,1\}^N} \left(\boldsymbol{x}^T \boldsymbol{Q} \boldsymbol{x} + \boldsymbol{d}^T \boldsymbol{x} \right) + \sum_{i=1}^{N} \mathbb{E}_{P_i} [\tilde{c}_i - d_i]^+ \right). \tag{2.4}$$

Then the optimal objective values of the two formulations are equal, $\beta^* = \beta^{**}$.

The proof of Proposition 2 is provided in the Appendix. Formulation (2.4) exploits the marginal specification of the joint distribution to provide a convex formulation in the variable $\boldsymbol{d}$. The objective function in (2.4) consists of two parts: (a) A deterministic QUBO problem with an objective of maximizing $\boldsymbol{x}^T \boldsymbol{Q} \boldsymbol{x} + \boldsymbol{d}^T \boldsymbol{x}$ for a fixed $\boldsymbol{d}$ and (b) A sum of N univariate convex penalty terms, each of the form $\mathbb{E}_{P_i}[\tilde{c}_i - d_i]^+$. The reformulation in (2.4) is NP-hard to solve since computing the first term in the objective for a fixed vector $\boldsymbol{d}$ is equivalent to solving a QUBO problem. A simple interpretation of this formulation is to find the balance between a deterministic approximation of the random QUBO problem based on the chosen $\boldsymbol{d}$ and a penalty term for choosing the vector $\boldsymbol{d}$ differently from the random vector $\tilde{\boldsymbol{c}}$. This result extends to discrete marginal distributions where in the proof, we need to replace the integrals with summations and use linear programming duality. It is also possible to extend the result of Proposition 2 to the case where only the mean and variance of each random variable is known. The result is stated in the next proposition.

Proposition 3. *Assume that the mean and variance for each $\tilde{c}_i$ are given , i.e.*

$$\mathbb{P}_i = \{P_i : \mathbb{E}_{P_i}(\tilde{c}_i) = \mu_i, \quad \mathbb{E}_{P_i}(\tilde{c}_i^2) = \mu_i^2 + \sigma_i^2\}, \; i \in [N].$$

Define

$$\beta^* := \sup_{P \in \mathbb{P}(\mathbb{P}_1,\ldots,\mathbb{P}_N)} \mathbb{E}_P \left[\max_{\boldsymbol{x} \in \{0,1\}^N} \left(\boldsymbol{x}^T \boldsymbol{Q} \boldsymbol{x} + \tilde{\boldsymbol{c}}^T \boldsymbol{x} \right) \right], \tag{2.5}$$

and

$$\beta^{**} = \min_{\boldsymbol{d} \in \Re^N} \left\{ \max_{\boldsymbol{x} \in \{0,1\}^N} (\boldsymbol{x}^T \boldsymbol{Q} \boldsymbol{x} + \boldsymbol{d}^T \boldsymbol{x}) + \sum_{i=1}^{N} \sup_{P_i \in \mathbb{P}_i} \mathbb{E}_{P_i} [\tilde{c}_i - d_i]^+ \right\}. \tag{2.6}$$

Then the optimal objective values of the formulations are equal, $\beta^* = \beta^{**}$.

The proof of Proposition 3 is provided in the Appendix.

3 The "Optimal" Preprocessing Vector

Our goal in this section is to find a preprocessing vector $\boldsymbol{u}$ such that the matrix $\boldsymbol{Q} - \mathrm{diag}(\boldsymbol{u})$ is negative semidefinite and it is an "optimal" choice for the extremal probability distribution of the random parameter vector $\tilde{\boldsymbol{c}}$ that attains the upper bound. Note that the "optimal" choice for $\boldsymbol{u}$ has to carefully defined for random QUBO problems since we are solving multiple instances of deterministic QUBO problems drawn from the extremal distribution.

Let $\mathbb{P}_i$ denote the set of possible marginal distributions for the random variable $\tilde{c}_i$. Assume that either the marginal distribution P_i of the random variable $\tilde{c}_i$ is given in which case the set $\mathbb{P}_i = \{P_i\}$ consists of a singleton or the mean and variance of $\tilde{c}_i$ is given in which case $\mathbb{P}_i = \{P_i : \mathbb{E}_{P_i}(\tilde{c}_i) = \mu_i, \ \mathbb{E}_{P_i}(\tilde{c}_i^2) = \mu_i^2 + \sigma_i^2\}$. Perturbing the objective function for the inner deterministic QUBO problem in the reformulations (2.4) or (2.6), we define:

$$\beta^*_{\boldsymbol{u}} \ = \ \min_{\boldsymbol{d} \in \Re^N} \left\{ \max_{\boldsymbol{x} \in \{0,1\}^N} \left[\boldsymbol{x}^T (\boldsymbol{Q} - \mathrm{diag}(\boldsymbol{u}))\boldsymbol{x} + (\boldsymbol{d} + \boldsymbol{u})^T \boldsymbol{x} \right] + \sum_{i=1}^N \sup_{P_i \in \mathbb{P}_i} \mathbb{E}_{P_i}[\tilde{c}_i - d_i]^+ \right\}, \tag{3.1}$$

From Propositions 2 and 3 and the observation that $\boldsymbol{x}^T \mathrm{diag}(\boldsymbol{u})\boldsymbol{x} = \boldsymbol{u}^T\boldsymbol{x}$ for $\boldsymbol{x} \in \{0,1\}^N$, the tight upper bound β^* is exactly equal to $\beta^*_{\boldsymbol{u}}$, namely:

$$\beta^* \ = \ \beta^*_{\boldsymbol{u}} \quad \forall \boldsymbol{u} \in \Re^N, \tag{3.2}$$

Define an upper bound $\bar{\beta}^*_{\boldsymbol{u}}$ on the optimal value β^* by using the continuous relaxation for the binary variables in the deterministic QUBO problem in (3.1):

$$\bar{\beta}^*_{\boldsymbol{u}} = \min_{\boldsymbol{d} \in \Re^N} \left\{ \max_{\boldsymbol{x} \in [0,1]^N} \left[\boldsymbol{x}^T (\boldsymbol{Q} - \mathrm{diag}(\boldsymbol{u}))\boldsymbol{x} + (\boldsymbol{d} + \boldsymbol{u})^T \boldsymbol{x} \right] + \sum_{i=1}^N \sup_{P_i \in \mathbb{P}_i} \mathbb{E}_{P_i}[\tilde{c}_i - d_i]^+ \right\}. \tag{3.3}$$

Then, clearly:

$$\beta^* \ \leq \ \bar{\beta}^*_{\boldsymbol{u}} \quad \forall \boldsymbol{u} \in \Re^N. \tag{3.4}$$

For a fixed perturbation vector $\boldsymbol{u}$ such that the matrix $\boldsymbol{Q} - \mathrm{diag}(\boldsymbol{u})$ is negative semidefinite, the objective function in $\bar{\beta}^*_{\boldsymbol{u}}$ is efficiently computable. This brings us to the definition of an "optimal" preprocessing vector for random QUBO problems.

Definition 1. *The "optimal" choice of the preprocessing vector for the random QUBO problem is defined as the vector* $\boldsymbol{u}^*_{\mathrm{opt}}$ *such that* $\boldsymbol{Q} - \mathrm{diag}(\boldsymbol{u})$ *is negative semidefinite and it minimizes the upper bound* $\bar{\beta}^*_{\boldsymbol{u}}$ *in (3.3) obtained from the continuous relaxation.*

In other words, $\boldsymbol{u}^*_{\text{opt}}$ is chosen to minimize the efficiently computable upper bound on the expectation for the random 0-1 quadratic programming problem obtained from the continuous relaxation:

$$\boldsymbol{u}^*_{\text{opt}} = \underset{diag(\boldsymbol{u})-\boldsymbol{Q}\succeq 0}{\text{argmin}} \min_{\boldsymbol{d}\in\Re^N} \left\{ \max_{\boldsymbol{x}\in[0,1]^N} \left[\boldsymbol{x}^T(\boldsymbol{Q} - \text{diag}(\boldsymbol{u}))\boldsymbol{x} + (\boldsymbol{d}+\boldsymbol{u})^T\boldsymbol{x}\right] + \sum_{i=1}^N \sup_{P_i\in\mathbb{P}_i} \mathbb{E}_{P_i}[\tilde{c}_i - d_i]^+ \right\}. \quad (3.5)$$

Define the smallest upper bound obtained from the continuous relaxation as:

$$\bar{\beta}^* = \bar{\beta}^*_{\boldsymbol{u}^*_{\text{opt}}}.$$

Then $\beta^* \leq \bar{\beta}^* \leq \bar{\beta}^*_{\boldsymbol{u}}$ for any $\boldsymbol{u}$. Changing the order of the minimization in the outer problems in (3.5), we get

$$\bar{\beta}^* = \min_{\boldsymbol{d}\in\Re^N} \left\{ \min_{diag(\boldsymbol{u})-\boldsymbol{Q}\succeq 0} \max_{\boldsymbol{x}\in[0,1]^N} \left[\boldsymbol{x}^T(\boldsymbol{Q} - \text{diag}(\boldsymbol{u}))\boldsymbol{x} + (\boldsymbol{d}+\boldsymbol{u})^T\boldsymbol{x}\right] + \sum_{i=1}^N \sup_{P_i\in\mathbb{P}_i} \mathbb{E}_{P_i}[\tilde{c}_i - d_i]^+ \right\}. \quad (3.6)$$

For a fixed vector $\boldsymbol{d}$, the inner subproblem:

$$\min_{diag(\boldsymbol{u})-\boldsymbol{Q}\succeq 0} \max_{\boldsymbol{x}\in[0,1]^N} [\boldsymbol{x}^T(\boldsymbol{Q} - \text{diag}(\boldsymbol{u}))\boldsymbol{x} + (\boldsymbol{d}+\boldsymbol{u})^T\boldsymbol{x}],$$

is solvable as a SDP using the same approach as for the deterministic QUBO (1.8). This brings us to the main result of the paper.

Proposition 4. *The upper bound $\bar{\beta}^*$ on the expected optimal objective value of a QUBO problem obtained from its convex relaxation in (3.6) is equal to the optimal value of the following SDP:*

$$\begin{aligned} \bar{\beta}^* = \quad & \min_{\boldsymbol{d},r,\boldsymbol{u}} \quad r + \sum_{i=1}^N \sup_{P_i\in\mathbb{P}_i} \mathbb{E}_{P_i}[\tilde{c}_i - d_i]^+ \\ & \quad s.t. \quad \begin{bmatrix} r & -(\boldsymbol{d}+\boldsymbol{u})^T/2 \\ -(\boldsymbol{d}+\boldsymbol{u})/2 & \text{diag}(\boldsymbol{u}) - \boldsymbol{Q} \end{bmatrix} \succeq 0. \end{aligned} \quad (3.7)$$

Furthermore the optimal decision vector $\boldsymbol{u}$ is $\boldsymbol{u}^*_{\text{opt}}$ which satisfies $\text{diag}(\boldsymbol{u}^*_{\text{opt}}) - \boldsymbol{Q} \succeq 0$ and $\bar{\beta}^* = \bar{\beta}^*_{\boldsymbol{u}^*_{\text{opt}}}$.

An alternate way to express formulation (3.7) is using the classical semidefinite relaxation of the deterministic QUBO problem as follows:

$$\begin{aligned} \bar{\beta}^* = \quad & \min_{\boldsymbol{d}\in\Re^N} \max_{\boldsymbol{x},\boldsymbol{X}} \quad \sum_{i=1}^N \sum_{j=1}^N Q_{ij}X_{ij} + \sum_{i=1}^N d_i x_i + \sum_{i=1}^N \sup_{P_i\in\mathbb{P}_i} \mathbb{E}_{P_i}[\tilde{c}_i - d_i]^+ \\ & \quad \text{s.t.} \quad \begin{bmatrix} 1 & \boldsymbol{x}^T \\ \boldsymbol{x} & \boldsymbol{X} \end{bmatrix} \succeq 0 \\ & \qquad\quad X_{ii} = x_i, \qquad\qquad i \in [N], \end{aligned} \quad (3.8)$$

where $\boldsymbol{u}_{\text{opt}}^*$ is the optimal dual variables to the equality constraints $X_{ii} = x_i,\ i = 1, \ldots, N$. The main difference is that the vector $\boldsymbol{d}$ is a decision variable with an additional penalty term that is separable across $i \in [N]$. The convex SDP formulation in (3.7) is a penalized version of the SDP in (1.8) where the penalty term is the sum of N univariate convex functions $\sup_{P_i \in \mathbb{P}_i} \mathbb{E}_{P_i}[\tilde{c}_i - d_i]^+$. Hence (3.7) can be interpreted as a Penalized QCR.

Consider a deterministic vector where $\tilde{\boldsymbol{c}} = \boldsymbol{c}$ with probability 1. We show that in this case, (3.7) reduces to (1.8). For a deterministic instance, formulation (3.7) reduces to

$$\begin{aligned} \bar{\beta}^* = \ & \min_{\boldsymbol{d}, r, \boldsymbol{u}} \ r + \sum_{i=1}^{N} [c_i - d_i]^+ \\ & \text{s.t.} \ \begin{bmatrix} r & -(\boldsymbol{d} + \boldsymbol{u})^T/2 \\ -(\boldsymbol{d} + \boldsymbol{u})/2 & \operatorname{diag}(\boldsymbol{u}) - \boldsymbol{Q} \end{bmatrix} \succeq 0. \end{aligned} \tag{3.9}$$

It is straightforward to verify that $\boldsymbol{d} = \boldsymbol{c}$ is optimal for (3.9). Notice that $\bar{\beta}^*$ is the optimal objective value to the problem:

$$\bar{\beta}^* = \min_{diag(\boldsymbol{u}) - \boldsymbol{Q} \succeq 0} \ \min_{\boldsymbol{d} \in \Re^N} \left\{ \max_{\boldsymbol{x} \in [0,1]^N} [\boldsymbol{x}^T (\boldsymbol{Q} - \operatorname{diag}(\boldsymbol{u})) \boldsymbol{x} + (\boldsymbol{d} + \boldsymbol{u})^T \boldsymbol{x}] + \sum_{i=1}^{N} [c_i - d_i]^+ \right\}. \tag{3.10}$$

Let $\boldsymbol{d}^*$ be the optimal vector in (3.10). If there exists some index i such that $d_i^* > c_i$, by setting $d_i = c_i$ the second term $\sum_{i=1}^N [c_i - d_i]^+$ in (3.10) will remain unchanged, while the first term $\max_{\boldsymbol{x} \in [0,1]^N} [\boldsymbol{x}^T (\boldsymbol{Q} - \operatorname{diag}(\boldsymbol{u})) \boldsymbol{x} + (\boldsymbol{d} + \boldsymbol{u})^T \boldsymbol{x}]$ will not increase. Similarly, if there exists some index i such that $d_i^* < c_i$, by setting $d_i = c_i$ the second term $\sum_{i=1}^N [c_i - d_i]^+$ in (3.10) will decrease by $c_i - d_i^*$, while the first term $\max_{\boldsymbol{x} \in [0,1]^N} [\boldsymbol{x}^T (\boldsymbol{Q} - \operatorname{diag}(\boldsymbol{u})) \boldsymbol{x} + (\boldsymbol{d} + \boldsymbol{u})^T \boldsymbol{x}]$ will increase by at most $c_i - d_i^*$. Hence $\boldsymbol{d} = \boldsymbol{c}$ is optimal for (3.9). Thus, the SDP reduces to the deterministic formulation (1.8).

Proposition 5. *Consider a deterministic vector where $\tilde{\boldsymbol{c}} = \boldsymbol{c}$ with probability 1. Then the SDP in (3.7) is equivalent to the SDP in (1.8).*

4 Computational Results

In this section, we apply the Penalized QCR method to solve a set of K quadratic unconstrained binary optimization problems where the instances are generated randomly from a probability distribution:

$$\beta(\boldsymbol{c}^{(k)}, \boldsymbol{Q}) = \max_{\boldsymbol{x} \in \{0,1\}^N} \left\{ q\left(\boldsymbol{x}; \boldsymbol{c}^{(k)}\right) := \boldsymbol{x}^T \boldsymbol{Q} \boldsymbol{x} + {\boldsymbol{c}^{(k)}}^T \boldsymbol{x} \right\}, \quad k \in [K]. \tag{4.1}$$

We solve the K instances of problem (4.1) using four different preprocessing approaches:

(a) **Eigenvalue based method:** In this method, we choose a common preprocessing vector by computing the maximum eigenvalue: $\boldsymbol{u}_{\text{eig}} = \lambda_{\max}(\boldsymbol{Q})\boldsymbol{e}$.

(b) **Sample based method:** In this method, we choose an optimal preprocessing vector $\boldsymbol{u}_{\text{opt}}^{(k)}$ for each instance $\boldsymbol{c} = \boldsymbol{c}^{(k)}$ by solving the semidefinite program (1.8). Thus we solve a total of K SDP problems.

(c) **Mean based method:** In this method, we choose a common preprocessing vector $\boldsymbol{u}_{\boldsymbol{\mu}}$ for $\boldsymbol{c} = \boldsymbol{\mu}$ by solving the semidefinite program (1.8). Thus we solve a single SDP.

(d) **Mean and standard deviation based method:** In this method, we choose a common preprocessing vector by solving a single semidefinite program for the Fréchet class of distributions. In our numerical experiments, we assume that only the mean μ_i and the standard deviation σ_i for each random variable is known. In this case, the penalty term $\sup_{P_i \in \mathbb{P}_i} \mathbb{E}[\tilde{c}_i - d_i]^+$ in the SDP (3.7) has a simple closed form expression based on the Cauchy-Schwarz inequality (see Scarf [37]):

$$\sup_{P_i \in \mathbb{P}_i} \mathbb{E}_{P_i}[\tilde{c}_i - d_i]^+ = \frac{1}{2}\left[(\mu_i - d_i) + \sqrt{(\mu_i - d_i)^2 + \sigma_i^2}\right].$$

Thus the preprocessing parameter $\boldsymbol{u}_{\boldsymbol{\mu},\boldsymbol{\sigma}}$ is obtained by solving the SDP:

$$\begin{aligned} \min_{\boldsymbol{d},r,\boldsymbol{u}} \quad & r + \frac{1}{2}\sum_{i=1}^{N}\left[(\mu_i - d_i) + \sqrt{(\mu_i - d_i)^2 + \sigma_i^2}\right] \\ \text{s.t.} \quad & \begin{bmatrix} r & -(\boldsymbol{d}+\boldsymbol{u})^T/2 \\ -(\boldsymbol{d}+\boldsymbol{u})/2 & \operatorname{diag}(\boldsymbol{u}) - \boldsymbol{Q} \end{bmatrix} \succeq 0. \end{aligned} \tag{4.2}$$

This is equivalent to the following SDP with one positive semidefinite matrix of size $(N+1)\times(N+1)$ and N second order conic programming (SOCP) constraints:

$$\begin{aligned} \min_{\boldsymbol{d},r,\boldsymbol{u},\boldsymbol{t}} \quad & r + \tfrac{1}{2}\left[\boldsymbol{e}^T(\boldsymbol{\mu} - \boldsymbol{d}) + \boldsymbol{e}^T\boldsymbol{t}\right] \\ \text{s.t.} \quad & \begin{bmatrix} r & -(\boldsymbol{d}+\boldsymbol{u})^T/2 \\ -(\boldsymbol{d}+\boldsymbol{u})/2 & \operatorname{diag}(\boldsymbol{u}) - \boldsymbol{Q} \end{bmatrix} \succeq 0, \\ & \left\| \begin{bmatrix} \mu_i - d_i \\ \sigma_i \end{bmatrix} \right\| \le t_i, \; i = 1, \ldots, N. \end{aligned} \tag{4.3}$$

Define the value $\text{obj}_k(\boldsymbol{c}^{(k)}, \boldsymbol{Q})$ as follows:

$$\text{obj}_k(\boldsymbol{c}^{(k)}, \boldsymbol{Q}) = \begin{cases} \beta(\boldsymbol{c}^{(k)}, \boldsymbol{Q}), & \text{if the QUBO problem is solvable within T minutes,} \\ \text{Best lower bound found,} & \text{otherwise.} \end{cases}$$

In our computational experiments, we set $T = 10$ minutes. We define $\text{gap}_k(\boldsymbol{u})$ as the relative difference between the objective function of the convex relaxation for a given preprocessing vector $\boldsymbol{u}$ and the value $\text{obj}_k(\boldsymbol{c}^{(k)}, \boldsymbol{Q})$:

$$\text{gap}_k(\boldsymbol{u}) = \frac{\bar{\beta}(\boldsymbol{u}; \boldsymbol{c}^{(k)}, \boldsymbol{Q}) - \text{obj}_k(\boldsymbol{c}^{(k)}, \boldsymbol{Q})}{\text{obj}_k(\boldsymbol{c}^{(k)}, \boldsymbol{Q})},$$

Since the running time of the branch-and-bound method to solve the binary quadratic program depends on the strength of its convex relaxation, we say that a vector $\boldsymbol{u}$ is preferable to $\boldsymbol{u}'$ for the kth instance if $\text{gap}_k(\boldsymbol{u}) < \text{gap}_k(\boldsymbol{u}')$. The computational studies were implemented in Matlab R2012a on an Intel Core 2 Duo CPU (2.8 GHz) laptop with 4 GB of RAM. The SDP problems were solved with CVX ([23, 22]) and SDPT3 ([40, 41]), and the 0-1 quadratic programming problems were solvd with CPLEX 12.4 using the Matlab interface.

4.1 Randomly Generated Instances

Given the mean $\boldsymbol{\mu}$ and the covariance matrix Σ, we generate the scenarios $\boldsymbol{c}^{(k)}$, $k \in [K]$ from a multivariate normal distribution $N(\boldsymbol{\mu}, \Sigma)$. The parameters are chosen in the following manner:

1. $\boldsymbol{Q}$ is a symmetric random matrix with density $d \in (0, 1]$. The density refers to the probability that an entry of $\boldsymbol{Q}$ is nonzero. Each nonzero entry is the sum of one or more normally distributed random variables.

2. Each component of the mean vector $\boldsymbol{\mu}$ is randomly generated from the standard normal distribution.

3. Each component of the vector of standard deviations $\boldsymbol{\sigma}$ is randomly generated from the uniform distribution $U(0, M)$, where M is a given positive number. The covariance matrix Σ is obtained from a randomly generated correlation matrix and the standard deviation vector $\boldsymbol{\sigma}$.

For a given pair of parameters (N, d) we generate the symmetric matrix $\boldsymbol{Q}$ of size $N \times N$ with density d and $K = 100$ instances of $\boldsymbol{c}^{(k)}$ from a normal distribution $N(\boldsymbol{\mu}, \Sigma)$. We compare the quality of the bounds and the CPU times to solve these instances with the four different choices of preprocessing vectors $\boldsymbol{u}$. In our computations, we allow for a maximum CPU time of 10 minutes to solve the binary quadratic program. The numerical results are shown in the Tables 1 and 2 where we set $M = 1$ and $M = 20$ respectively. In these Tables, we report the following values for the four different choices of preprocessing vectors $\boldsymbol{u} = \boldsymbol{u}_{\text{eig}}$, $\boldsymbol{u} = \boldsymbol{u}_{\text{opt}}^{(k)}$, $\boldsymbol{u} = \boldsymbol{u}_{\mu}$ and $\boldsymbol{u} = \boldsymbol{u}_{\boldsymbol{\mu},\boldsymbol{\sigma}}$:

1. The average gap over the 100 instances which is given as gap $= \sum_{k=1}^{100} \text{gap}_k(\boldsymbol{u})/100$.
2. The CPU time taken to compute the preprocessing parameter $\boldsymbol{u}$ denoted by "t_$\boldsymbol{u}$". For the sample based method, "t_$\boldsymbol{u}$" is the total CPU time taken to solve the 100 SDPs. For the mean based and mean and standard deviation based methods, "t_$\boldsymbol{u}$" is the CPU time taken to solve a single SDP. For the eigenvalue based method, "t_$\boldsymbol{u}$" is the CPU time taken to compute the largest eigenvalue of $\boldsymbol{Q}$.
3. The total CPU time taken to compute all the QUBO problems solvable by CPLEX within 10 minutes for a given preprocessing parameter $\boldsymbol{u}$. This is denoted by "t_01QP". If we solve every instance within 10 minutes, we report the total CPU time. If there are $m < 100$ instances that are solvable within 10 minutes each, we report the total CPU time to solve these m instances and report the average time for the m solved instances in the parentheses.
4. The number of instances (out of 100) which are solved within 10 minutes is denoted by "solved".

From Table 1, we observe that when the standard deviation is small ($\boldsymbol{\sigma} = rand(N, 1)$), the relative gaps for $\boldsymbol{u}_{\boldsymbol{\mu}}$ and $\boldsymbol{u}_{\boldsymbol{\mu},\boldsymbol{\sigma}}$ are much smaller than the relative gap for $\boldsymbol{u}_{\text{eig}}$, and very close to the relative gap for the sample based method. Although the preprocessing parameter $\boldsymbol{u}_{\text{eig}}$ can be computed very efficiently, solving the QUBO problem is much slower than the other three methods. Since the standard deviation is of a similar magnitude as the mean, $\boldsymbol{u}_{\boldsymbol{\mu}}$ and $\boldsymbol{u}_{\boldsymbol{\mu},\boldsymbol{\sigma}}$ have similar relative gaps. These two methods are also much faster than finding the preprocessing step for the sample based method that involves solving 100 SDP instances.

From Table 2, we observe that when the standard deviation is larger ($\boldsymbol{\sigma} = 20 * rand(N, 1)$), the relative gap from the sample based method is much smaller than the gaps generated from the other three methods. In these cases, we have $\text{gap}(\boldsymbol{u}_{\text{opt}}^{(k)}) < \text{gap}(\boldsymbol{u}_{\boldsymbol{\mu},\boldsymbol{\sigma}}) < \text{gap}(\boldsymbol{u}_{\boldsymbol{\mu}}) < \text{gap}(\boldsymbol{u}_{\text{eig}})$. The CPU time taken to solve the QUBO problem by choosing $\boldsymbol{u}_{\boldsymbol{\mu},\boldsymbol{\sigma}}$ is smaller than that by choosing $\boldsymbol{u}_{\text{eig}}$ and $\boldsymbol{u}_{\boldsymbol{\mu}}$, and it is close to the CPU time taken by using $\boldsymbol{u}_{\text{opt}}^{(k)}$. Lastly, the computational time needed for the preprocessing step for the mean and standard deviation based method is much lesser than that for the preprocessing step of the sample based method. As a result, the total CPU time needed to solve all the 100 QUBO problems to optimality using the preprocessing vector $\boldsymbol{u}_{\boldsymbol{\mu},\boldsymbol{\sigma}}$ is substantially smaller than that needed by the sample based preprocessing method.

Table 1: Gap and CPU time for different parameters $\boldsymbol{u}$ when $\boldsymbol{\mu} = randn(N, 1), \boldsymbol{\sigma} = rand(N, 1)$

N	d	$\boldsymbol{u} = \boldsymbol{u}_{\text{eig}}$				$\boldsymbol{u} = \boldsymbol{u}_{\text{opt}}^{(k)}, k \in [K]$				$\boldsymbol{u} = \boldsymbol{u}_{\boldsymbol{\mu}}$				$\boldsymbol{u} = \boldsymbol{u}_{\boldsymbol{\mu},\boldsymbol{\sigma}}$			
		gap	t_$\boldsymbol{u}$	t_01QP	solved	gap	t_$\boldsymbol{u}$	t_01QP	solved	gap	t_$\boldsymbol{u}$	t_01QP	solved	gap	t_$\boldsymbol{u}$	t_01QP	solved
50	0.4	18.5	0.01	101.7	100	5.0	48.76	15.3	100	5.4	0.47	15.0	100	5.6	1.64	12.8	100
50	0.6	13.6	0.02	50.7	100	3.5	45.57	13.2	100	3.9	0.44	13.5	100	4.0	1.66	12.5	100
50	1.0	18.6	0.02	196.9	100	8.4	46.53	32.8	100	8.9	0.46	30.3	100	8.9	1.53	26.3	100
60	0.2	12.0	0.02	161.8	100	2.7	52.46	14.6	100	3.1	0.67	14.3	100	3.2	1.79	12.0	100
60	0.4	22.2	0.02	3599.7	100	8.2	52.91	69.3	100	9.0	0.64	59.4	100	9.0	1.76	45.5	100
70	0.3	19.4	0.02	6355.1	100	4.9	59.50	51.8	100	5.5	0.79	49.9	100	5.5	2.06	35.6	100
70	0.8	19.3	0.02	19955.2	100	8.1	55.91	246.0	100	8.4	0.72	277.8	100	8.5	2.09	223.9	100
80	0.2	23.0	0.02	5504.5(458.7)	12	6.5	65.44	136.8	100	7.5	1.00	140.9	100	7.5	2.54	87.6	100
80	0.7	14.9	0.02	8469.4	100	6.5	67.46	334.0	100	6.8	0.86	327.2	100	6.9	2.82	318.2	100
90	0.3	19.9	0.02	**	0	7.2	90.18	899.0	100	7.6	1.03	923.0	100	7.7	3.55	830.6	100
100	0.1	20.1	0.02	**	0	4.7	103.20	229.3	100	5.7	1.26	238.7	100	5.7	4.09	175.0	100

Table 2: Gap and CPU time for different parameters $\boldsymbol{u}$ when $\boldsymbol{\mu} = randn(N, 1), \boldsymbol{\sigma} = 20 * rand(N, 1)$

N	d	$\boldsymbol{u} = \boldsymbol{u}_{\text{eig}}$				$\boldsymbol{u} = \boldsymbol{u}_{\text{opt}}^{(k)}, k \in [K]$				$\boldsymbol{u} = \boldsymbol{u}_{\boldsymbol{\mu}}$				$\boldsymbol{u} = \boldsymbol{u}_{\boldsymbol{\mu},\boldsymbol{\sigma}}$			
		gap	t_$\boldsymbol{u}$	t_01QP	solved	gap	t_$\boldsymbol{u}$	t_01QP	solved	gap	t_$\boldsymbol{u}$	t_01QP	solved	gap	t_$\boldsymbol{u}$	t_01QP	solved
50	0.4	10.0	0.01	18.9	100	2.6	46.19	9.2	100	8.0	0.46	9.5	100	5.5	1.54	9.3	100
50	0.6	9.4	0.02	27.9	100	2.5	45.24	11.9	100	8.4	0.44	13.2	100	5.7	1.48	11.9	100
50	1.0	11.6	0.01	70.0	100	4.7	45.88	20.6	100	10.6	0.45	21.7	100	8.2	1.52	19.2	100
60	0.2	5.6	0.02	6.3	100	0.8	54.04	6.2	100	5.1	0.59	6.2	100	2.7	1.78	6.1	100
60	0.4	8.5	0.02	39.3	100	2.4	53.68	11.9	100	6.9	0.61	13.4	100	5.6	1.79	12.8	100
70	0.3	8.5	0.02	100.4	100	2.0	61.98	17.1	100	7.3	1.08	23.5	100	4.8	2.15	20.7	100
70	0.8	12.2	0.02	2097.2(21.2)	99	4.0	60.15	80.5	100	9.3	0.89	115.7	100	7.1	2.07	105.7	100
80	0.2	6.9	0.02	203.5	100	1.4	77.65	12.6	100	5.4	0.90	19.4	100	3.9	3.01	17.4	100
80	0.7	9.3	0.02	2373.9	100	3.6	68.24	126.8	100	8.7	0.66	255.6	100	6.7	2.41	151.2	100
90	0.3	8.6	0.02	4638.1	100	2.6	85.46	88.9	100	7.0	0.83	215.3	100	5.7	2.76	170.9	100
100	0.1	6.6	0.02	39.7	100	1.2	99.15	9.9	100	5.3	0.94	12.9	100	3.6	3.11	12.3	100

4.2 Instances from Billionnet and Elloumi [8] and Pardalos and Rodgers [33]

We use the set of randomly generated instances as in Billionnet and Elloumi [8] and Pardalos and Rodgers [33]. We choose the parameters as follows:

1. The linear coefficients c_i are chosen uniformly and independently in the range $[-100, 100]$.
2. The diagonal entries of $\boldsymbol{Q} \in \Re^{N \times N}$ are all 0, and the off-diagonal coefficients of the symmetric matrix $\boldsymbol{Q}$ are in the range $[-50, 50]$.
3. The matrix $\boldsymbol{Q}$ has density d. The density refers to the probability that a nonzero will occur in any off-diagonal entry.

In this example, the data $\boldsymbol{c}^{(k)}$ for $K = 100$ samples are given. We use the sample mean and the sample standard deviation to compute the preprocessing parameters $\boldsymbol{u}_{\boldsymbol{\mu}}$ and $\boldsymbol{u}_{\boldsymbol{\mu},\boldsymbol{\sigma}}$. Again, we use the four different preprocessing methods to solve the QUBO problems, and the maximum CPU time taken to solve the QUBO problem is set to be 10 minutes. The results are listed in Table 3. In addition to the

Table 3: Gap and CPU time for different parameters $\boldsymbol{u}$

N	d	$\boldsymbol{u} = \boldsymbol{u}_{\text{eig}}$				$\boldsymbol{u} = \boldsymbol{u}_{\text{opt}}^{(k)}, k \in [K]$				$\boldsymbol{u} = \boldsymbol{u}_{\boldsymbol{\mu}}$				$\boldsymbol{u} = \boldsymbol{u}_{\boldsymbol{\mu},\boldsymbol{\sigma}}$			
		gap	t_$\boldsymbol{u}$	t_01QP	solved	gap	t_$\boldsymbol{u}$	t_01QP	solved	gap	t_$\boldsymbol{u}$	t_01QP	solved	gap	t_$\boldsymbol{u}$	t_01QP	solved
50	0.4	13.8	0.02	106.0	100	4.9	45.27	23.0	100	8.9	0.48	20.5	100	7.3	1.52	18.0	100
50	0.6	15.1	0.01	109.0	100	6.7	45.39	28.2	100	10.0	0.46	24.3	100	9.0	1.55	19.4	100
50	1.0	12.5	0.02	85.9	100	6.6	45.84	30.9	100	8.8	0.45	26.0	100	8.5	1.55	24.1	100
60	0.2	17.2	0.01	1022.4	100	5.8	53.90	43.5	100	10.7	0.63	53.6	100	8.7	1.82	35.8	100
60	0.4	14.0	0.01	1136.3	100	4.4	51.96	50.8	100	7.1	0.50	42.9	100	6.1	1.78	37.1	100
70	0.3	18.5	0.02	1249.4	100	8.2	58.50	280.0	100	12.2	0.69	292.5	100	10.9	2.20	261.5	100
80	0.2	18.4	0.03	16569.2(218.0)	76	7.8	70.68	450.9	100	12.4	0.67	545.1	100	10.8	2.40	435.3	100
90	0.6	18.3	0.03	13764.3(327.7)	42	9.1	83.60	7040.1	100	11.4	0.83	7095.1	100	11.0	3.26	6783.9	100
100	0.1	16.5	0.02	4761.2(297.6)	16	3.5	100.83	178.2	100	7.9	1.21	317.1	100	5.9	4.05	210.7	100
120	0.2	21.1	0.05	**	0	10.4	129.52	5770.7(360.7)	16	13.6	1.35	3610.4(361.0)	10	13.0	4.03	5500.0(343.8)	16

average gap, we plot the distributions of the relative gaps for the 100 scenarios using the boxplot in Figure 1. From the results, we observe that the average relative gap of using $\boldsymbol{u}_{\boldsymbol{\mu},\boldsymbol{\sigma}}$ is always smaller than using $\boldsymbol{u}_{\text{eig}}$ and $\boldsymbol{u}_{\boldsymbol{\mu}}$. In addition to the average value, from Figure 1 we observe that the relative gap of using $\boldsymbol{u}_{\boldsymbol{\mu},\boldsymbol{\sigma}}$ has a smaller sample minimum, lower quartile (25th percentile), median, upper quartile (75th percentile), and sample maximum than using $\boldsymbol{u}_{\text{eig}}$ and $\boldsymbol{u}_{\boldsymbol{\mu}}$. The relative gap using the sample based method is the smallest as should be expected. Hence, in terms of the relative gap between the optimal value of the QUBO problem and its convex relaxation, parameter $\boldsymbol{u}_{\boldsymbol{\mu},\boldsymbol{\sigma}}$ is better than $\boldsymbol{u}_{\text{eig}}$ and $\boldsymbol{u}_{\boldsymbol{\mu}}$ and closest to the sample based method.

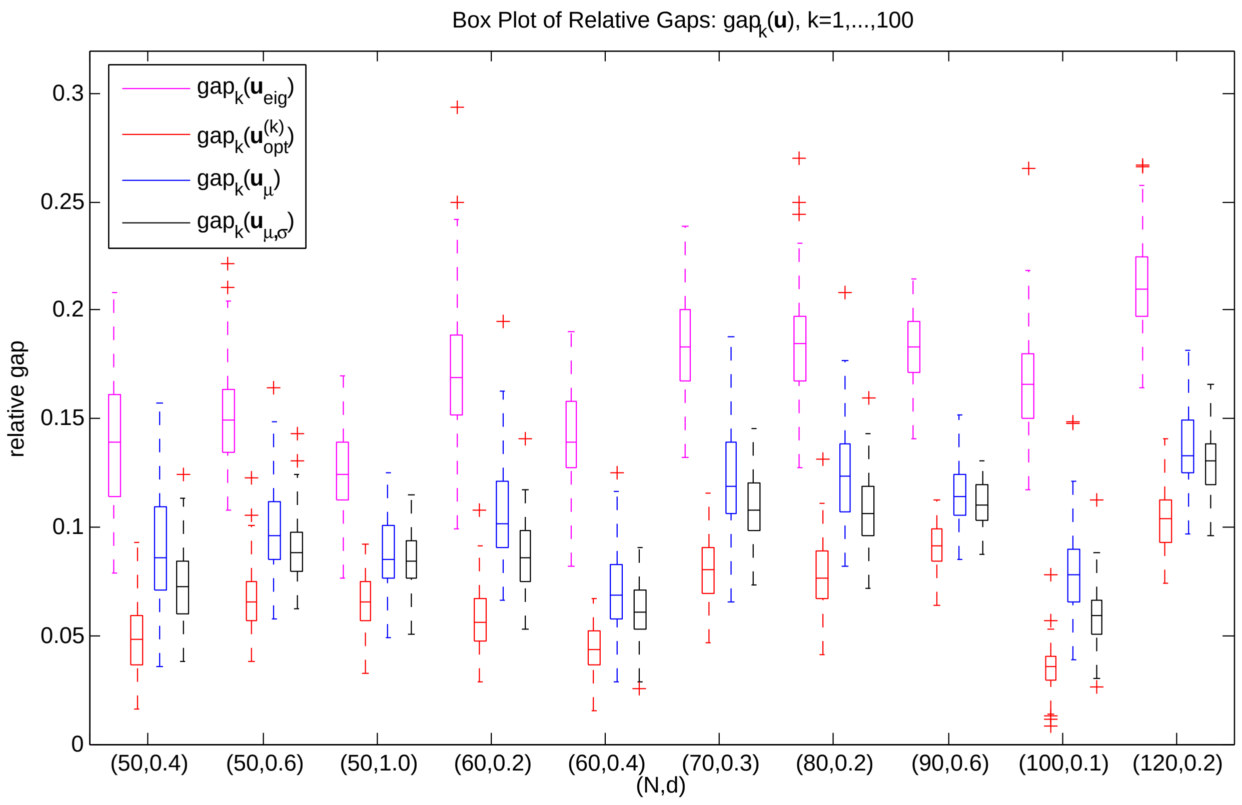


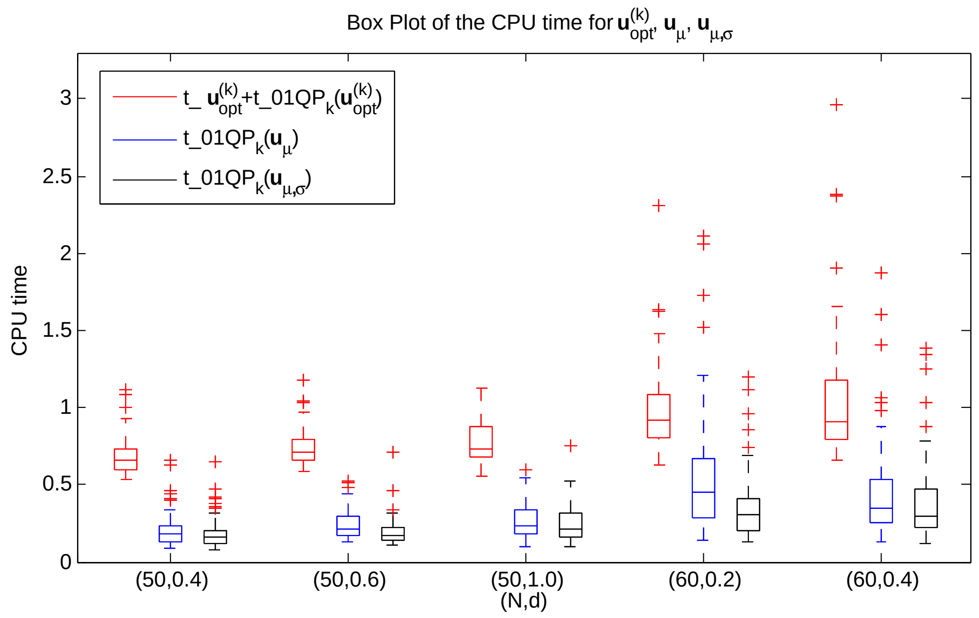
Figure 1: Boxplot of the Relative Gaps for all the 100 scenarios

We also plot the CPU time taken to solve the QUBO problem for every scenario $\boldsymbol{c}^{(k)}$, $k \in [K]$. In Figure 2, "t_01QP$_k(\boldsymbol{u})$", denotes the CPU time taken to solve the convex QUBO problem with the preprocessing parameter $\boldsymbol{u}$ for scenario $\boldsymbol{c}^{(k)}$. Since the CPU time taken to solve the QUBO problem by using $\boldsymbol{u}_{\text{eig}}$ is much larger than the other three methods, we exclude the eigenvalue based method from consideration. Since $\boldsymbol{u}_{\boldsymbol{\mu}}$ and $\boldsymbol{u}_{\boldsymbol{\mu},\boldsymbol{\sigma}}$ are common preprocessing parameters for all the 100 instance, and we can compute them quickly by solving a single SDP problem, the CPU time of getting $\boldsymbol{u}_{\boldsymbol{\mu}}$ and $\boldsymbol{u}_{\boldsymbol{\mu},\boldsymbol{\sigma}}$ is negligible in Figure 2. However, to use $\{\boldsymbol{u}^{(k)}_{\text{opt}},\ k \in [K]\}$ we must solve an SDP problem for every instance. Hence in the plot of the CPU time to solve the QUBO problem, the time (t_$\boldsymbol{u}^{(k)}_{\text{opt}}$) taken to compute $\boldsymbol{u}^{(k)}_{\text{opt}}$ is added.

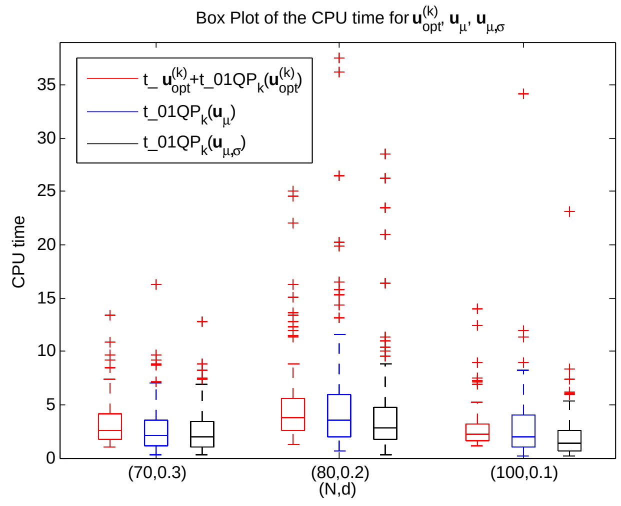
From Figure 2, we see that for small size instances (subfigure (a)) and medium size instances (subfigure (b)), the mean and standard deviation method is better than the sample based and mean based method. The CPU time of using $\boldsymbol{u}_{\boldsymbol{\mu},\boldsymbol{\sigma}}$ to solve the QUBO problem have the smallest sample minimum, lower quartile (25th percentile), median, upper quartile (75th percentile), and sample maximum. For the difficult instances (subfigure (c)), the three methods look more similar in Figure 2. From Table 3, we can see that using $\boldsymbol{u}_{\boldsymbol{\mu},\boldsymbol{\sigma}}$, we need the smallest CPU time to solve all the 100 instances when $N = 90, d = 0.6$. For the largest and most difficult set of instances with $N = 120, d = 0.2$, very few instances can be solved to optimality in 10 minutes. By using $\boldsymbol{u}_{\boldsymbol{\mu},\boldsymbol{\sigma}}$, we solve 16 instances to optimality which is the same as using the sample based method.

Robustness Tests using Permutations

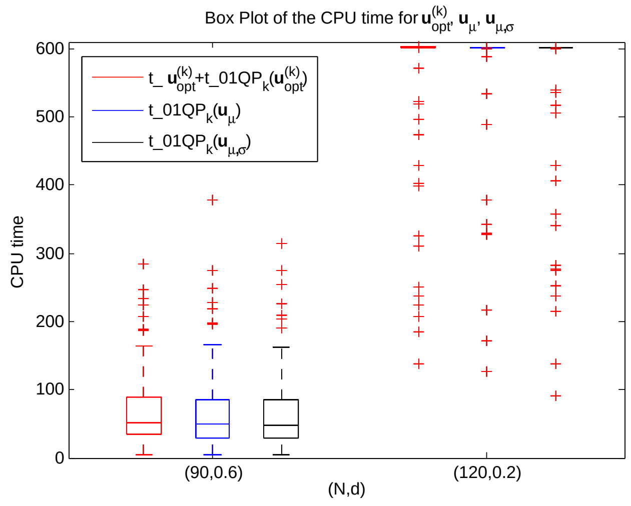
In this section, we test the robustness of the mean and standard deviation based method using permutation experiments. The Penalized QCR method in this paper is developed for the Fréchet class of distributions with fixed marginal distributions. However no assumption is made on the dependency structure between random variables. To test the robustness of the solutions, we generate other feasible distributions in this set by permuting the individual components of the randomly generated samples in the following manner. Given the sample data $\boldsymbol{c}^{(1)}, \ldots, \boldsymbol{c}^{(100)}$, we compute the sample mean $\boldsymbol{\mu}$ and the sample standard deviation $\boldsymbol{\sigma}$. For $i \in [N]$, we randomly permute the ith component sequence of the vectors $c_i^{(1)}, c_i^{(2)}, \ldots, c_i^{(100)}$. By performing this permutation independently for each $i \in [N]$, we generate a new set of samples $\{\bar{\boldsymbol{c}}^{(k)}, k \in [K]\}$. See [32] for a similar set of experiments in the context of stochastic knapsack problems. Note that the sample mean $\boldsymbol{\mu}$ and the standard deviation $\boldsymbol{\sigma}$ will not change after these permutations. Hence the preprocessing parameter $\boldsymbol{u}_{\boldsymbol{\mu}}$ and $\boldsymbol{u}_{\boldsymbol{\mu},\boldsymbol{\sigma}}$ will not change. However clearly, $\boldsymbol{u}^{(k)}_{\text{opt}}$ might change since the samples have changed. As a control, we also use the average sample based



(a) Small Size Instances



(b) Medium Size Instances



(c) Hard to Solve Instances

Figure 2: Boxplot of the CPU Time: (for the instances which can not be solved in 10 minutes, we just plot its CPU time as 600 seconds in the figure)

preprocessing vector defined as $\boldsymbol{u}_{\text{ave}} := \sum_{k=1}^{100} \boldsymbol{u}_{\text{opt}}^{(k)}/100$. Since $\boldsymbol{Q} - \text{diag}(\boldsymbol{u}_{\text{opt}}^{(k)}) \succeq 0, \forall k \in [K]$, we have $\boldsymbol{Q} - \text{diag}(\boldsymbol{u}_{\text{ave}}) \succeq 0$.

For the tests, we use two sets of parameters ($(N, d) = (50, 0.6)$ and $(N, d) = (70, 0.3)$) from Table 3 and perform numerical tests for the samples after the random permutations. For each set of data, we test the results across 15 permutations. The preprocessing vectors $\boldsymbol{u}_{\text{ave}}$, $\boldsymbol{u}_{\boldsymbol{\mu}}$ and $\boldsymbol{u}_{\boldsymbol{\mu},\boldsymbol{\sigma}}$ are computed only once and hence the CPU time of computing these preprocessing parameters is ignored. The numerical results are shown in Table 4 and 5.

From Tables 4 and 5, we see that by using $\boldsymbol{u}_{\boldsymbol{\mu},\boldsymbol{\sigma}}$ the average gap is smaller than using $\boldsymbol{u}_{\boldsymbol{\mu}}$ and $\boldsymbol{u}_{\text{ave}}$. Moreover the total CPU time taken to solve the QUBO problem is always the smallest for all the permutations by using $\boldsymbol{u}_{\boldsymbol{\mu},\boldsymbol{\sigma}}$. This shows that the mean and standard deviation based penalized QCR method is robust for the small and medium size instances.

5 Conclusions

In this paper we have developed a Penalized QCR method to solve random QUBO problems. The formulation can be viewed as a penalized version of the SDP used for deterministic QUBO problems. Using this SDP formulation, we find a common preprocessing vector for a random set of instances which differs only in the linear term of the objective. Computationally, we show that by using limited probabilistic information such as the mean and variance, and solving a single SDP across random instances of the problem, we obtain significant computational advantages over alternative preprocessing methods. Our results are developed for random QUBO problems where the probability distribution for the random linear term comes from the Fréchet class of distributions. Future research questions are to develop preprocessing methods for other representations of probability distributions and to generalize the method to random $\boldsymbol{Q}$ matrices.

References

[1] B. Alidaee, G. A. Kochenberger, and A. Ahmadian. 0-1 quadratic programming approach for optimum solutions of two scheduling problems. *International Journal of Systems Science*, 25(2):401–408, 1994.

[2] K. Allemand, K. Fukuda, T. M. Liebling, and E. Steiner. A polynomial case of unconstrained zero-one quadratic optimization. *Mathematical Programming*, 91(1):49–52, 2001.

[3] F. Barahona. A solvable case of quadratic 0–1 programming. *Discrete Applied Mathematics*, 13(1):23–26, 1986.

Table 4: Gap and CPU time with 15 permutations: $N = 50$, $d = 0.6$

No.	$\boldsymbol{u} = \boldsymbol{u}_{\text{opt}}^{(k)}, k \in [K]$				$\boldsymbol{u} = \boldsymbol{u}_{\text{ave}}$			$\boldsymbol{u} = \boldsymbol{u}_{\boldsymbol{\mu}}$			$\boldsymbol{u} = \boldsymbol{u}_{\boldsymbol{\mu},\boldsymbol{\sigma}}$		
	gap	t_$\boldsymbol{u}$	t_01QP	solved	gap	t_01QP	solved	gap	t_01QP	solved	gap	t_01QP	solved
1	6.6	47.78	22.6	100	10.5	22.1	100	10.0	19.7	100	8.9	15.2	100
2	6.5	55.79	27.9	100	10.4	25.3	100	9.9	23.9	100	8.8	18.7	100
3	6.7	56.25	29.6	100	10.6	27.7	100	10.1	25.8	100	9.0	19.8	100
4	6.7	58.73	27.7	100	10.7	26.5	100	10.1	24.5	100	9.0	19.1	100
5	6.5	53.95	27.8	100	10.4	26.3	100	9.9	24.2	100	8.8	19.0	100
6	6.5	54.24	27.3	100	10.4	25.6	100	9.9	23.8	100	8.8	18.8	100
7	6.7	56.78	28.4	100	10.9	27.8	100	10.4	25.7	100	9.1	19.5	100
8	6.6	54.65	27.2	100	10.5	26.0	100	9.9	23.8	100	8.9	18.7	100
9	6.5	55.60	26.7	100	10.4	24.7	100	9.9	23.4	100	8.8	18.3	100
10	6.5	45.54	26.7	100	10.4	25.0	100	9.9	23.1	100	8.8	18.0	100
11	6.6	52.42	27.4	100	10.5	26.6	100	10.0	24.7	100	8.9	18.9	100
12	6.5	52.29	27.4	100	10.4	26.2	100	9.8	23.5	100	8.8	18.8	100
13	6.7	54.48	28.3	100	10.6	26.8	100	10.1	24.9	100	9.0	19.4	100
14	6.6	53.38	26.9	100	10.5	25.0	100	10.0	22.8	100	8.9	18.3	100
15	6.4	56.66	24.5	100	10.3	22.8	100	9.8	21.5	100	8.7	17.0	100

Table 5: Gap and CPU time with 15 permutations: $N = 70$, $d = 0.3$

No.	$\boldsymbol{u} = \boldsymbol{u}_{\text{opt}}^{(k)}, k \in [K]$				$\boldsymbol{u} = \boldsymbol{u}_{\text{ave}}$			$\boldsymbol{u} = \boldsymbol{u}_{\boldsymbol{\mu}}$			$\boldsymbol{u} = \boldsymbol{u}_{\boldsymbol{\mu},\boldsymbol{\sigma}}$		
	gap	t_$\boldsymbol{u}$	t_01QP	solved	gap	t_01QP	solved	gap	t_01QP	solved	gap	t_01QP	solved
1	8.3	62.42	303.7	100	13.1	500.2	100	12.3	322.9	100	11.0	278.3	100
2	8.3	63.63	275.1	100	13.1	498.1	100	12.2	300.0	100	11.0	267.7	100
3	8.5	62.99	340.7	100	13.4	580.9	100	12.6	368.2	100	11.3	328.8	100
4	8.3	62.75	275.3	100	13.1	481.2	100	12.2	293.1	100	11.0	267.7	100
5	8.2	62.87	291.5	100	13.0	489.0	100	12.2	301.8	100	10.9	267.4	100
6	8.4	63.07	339.4	100	13.1	600.4	100	12.2	350.8	100	11.0	321.6	100
7	8.2	62.71	278.8	100	13.1	453.9	100	12.3	288.4	100	11.0	265.8	100
8	8.2	62.96	271.5	100	13.0	471.8	100	12.2	286.9	100	10.9	258.2	100
9	8.2	62.95	274.4	100	13.1	493.4	100	12.2	307.3	100	10.9	264.0	100
10	8.1	62.76	246.7	100	12.9	423.5	100	12.0	264.0	100	10.8	237.4	100
11	8.3	63.12	276.3	100	13.2	471.6	100	12.4	297.0	100	11.0	261.6	100
12	8.2	63.09	279.6	100	13.1	501.2	100	12.2	302.1	100	10.9	266.1	100
13	8.2	63.15	282.8	100	13.0	485.3	100	12.2	302.2	100	10.9	266.5	100
14	8.5	63.37	330.5	100	13.4	579.2	100	12.6	350.0	100	11.2	311.7	100
15	8.2	66.04	293.7	100	13.0	497.6	100	12.2	311.1	100	10.9	274.0	100

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

[5] J. E. Beasley. Heuristic algorithms for the unconstrained binary quadratic programming problem. *London, UK: Management School, Imperial College*, 4, 1998.

[6] D. Bertsimas, K. Natarajan, and C. P. Teo. Probabilistic combinatorial optimization: Moments, semidefinite programming, and asymptotic bounds. *SIAM Journal on Optimization*, 15(1):185–209, 2004.

[7] D. Bertsimas, K. Natarajan, and C. P. Teo. Persistence in discrete optimization under data uncertainty. *Mathematical Programming*, 108(2-3):251–274, 2006.

[8] A. Billionnet and S. Elloumi. Using a mixed integer quadratic programming solver for the unconstrained quadratic 0-1 problem. *Mathematical Programming*, 109(1):55–68, 2007.

[9] A. Billionnet, S. Elloumi, and A. Lambert. A branch and bound algorithm for general mixed-integer quadratic programs based on quadratic convex relaxation. *To appear in Journal of Combinatorial Optimization*, 2012.

[10] A. Billionnet, S. Elloumi, and A. Lambert. An efficient compact quadratic convex reformulation for general integer quadratic programs. *To appear in Computational Optimization and Applications*, 2012.

[11] A. Billionnet, S. Elloumi, and A. Lambert. Extending the QCR method to general mixed-integer programs. *Mathematical Programming*, 131(1-2):381–401, 2012.

[12] A. Billionnet, S. Elloumi, and M-C. Plateau. Improving the performance of standard solvers for quadratic 0-1 programs by a tight convex reformulation: The qcr method. *Discrete Applied Mathematics*, 157(6):1185–1197, 2009.

[13] J. R. Birge and M. J. Maddox. Bounds on expected project tardiness. *Operations Research*, 43:838–850, 1995.

[14] E. Boros and P. L. Hammer. The max-cut problem and quadratic 0-1 optimization; polyhedral aspects, relaxations and bounds. *Annals of Operations Research*, 33(3):151–180, 1991.

[15] M. W. Carter. The indefinite zero-one quadratic problem. *Discrete Applied Mathematics*, 7(1):23–44, 1984.

[16] G. Dall´ Aglio. Fréchet classes and compatibility of distribution functions. *Symposia Math.*, 9:131–150, 1972.

[17] L. Duan, S. Xiaoling, G. Shenshen, G. Jianjun, and Chunli. L. Polynomially solvable cases of binary quadratic programs. *Optimization and Optimal Control, Springer Optimization and Its Applications*, pages 199–225, 2010.

[18] A. Galli and A. N. Letchford. A compact variant of the qcr method for 0-1 quadratically constrained quadratic programs. *Working Paper*, 2013.

[19] M. R. Garey and D. S. Johnson. *Computers and intractability*, volume 174. 1979.

[20] B. Ghaddar, J. C. Vera, and M. F. Anjos. Second-order cone relaxations for binary quadratic polynomial programs. *SIAM Journal of Optimization*, 21(1):391–414, 2011.

[21] F. Glover, G. A. Kochenberger, and B. Alidaee. Adaptive memory search for binary quadratic programs. *Management Science*, 44(3):336–345, 1998.

[22] M. Grant and S. Boyd. Graph implementations for nonsmooth convex programs. *Recent Advances in Learning and Control (a tribute to M. Vidyasagar), V. Blondel, S. Boyd, and H. Kimura, editors, Lecture Notes in Control and Information Sciences*, pages 95–110, 2008.

[23] M. Grant and S. Boyd. Cvx research, inc. cvx: Matlab software for disciplined convex programming, version 2.0. beta. 2012.

[24] P. L. Hammer and A. A Rubin. Some remarks on quadratic programming with 0-1 variables. *Revue Francaise Informatique et de Recherche Operationnelle*, 4(3):67–79, 1970.

[25] C. Helmberg and F. Rendl. Solving quadratic (0, 1)-problems by semidefinite programs and cutting planes. *Mathematical Programming*, 82(3):291–315, 1998.

[26] M. Jünger, A. Martin, G. Reinelt, and R. Weismantel. Quadratic 0/1 optimization and a decomposition approach for the placement of electronic circuits. *Mathematical Programming*, 63(1):257–279, 1994.

[27] S. Kim and M. Kojima. Second order cone programming relaxation of nonconvex quadratic optimization problems. *Optimization Methods and Software*, 15(3-4):201–224, 2001.

[28] W. K. Klein Haneveld. Robustness against dependence in pert: An application of duality and distributions with known marginals. *Mathematical Programming Study*, 27:153–182, 1986.

[29] F. Körner. A tight bound for the boolean quadratic optimization problem and its use in a branch and bound algorithm. *Optimization*, 19(5):711–721, 1988.

[30] I. Meilijson and A. Nadas. Convex majorization with an application to the length of critical path. *Journal of Applied Probability*, 16:671–677, 1979.

[31] M. Muramatsu and T. Suzuki. A new second-order cone programming relaxation for max-cut problems. *Journal of Operations Research of Japan*, 43:164–177, 2003.

[32] K. Natarajan, M. Song, and C. P. Teo. Persistency model and its applications in choice modeling. *Management Science*, 55(3):453–469, 2009.

[33] P. M. Pardalos and G. P. Rodgers. Computational aspects of a branch and bound algorithm for quadratic zero-one programming. *Computing*, 45(2):131–144, 1990.

[34] J. C. Picard and H. D. Ratliff. Minimum cuts and related problems. *Networks*, 5(4):357–370, 1975.

[35] S. Poljak, F. Rendl, and H. Wolkowicz. A recipe for semidefinite relaxation for (0, 1)-quadratic programming. *Journal of Global Optimization*, 7(1):51–73, 1995.

[36] R. T. Rockafellar. Saddle points and convex analysis. *Differential games and related topics,H.W. Kuhn and G.P. Szego, eds.*, pages 109–128, 1971.

[37] H. Scarf. A min-max solution of an inventory problem. *Studies in The Mathematical Theory of Inventory and Production*, pages 201–209, 1958.

[38] N. Z Shor. Class of global minimum bounds of polynomial functions. *Cybernetics and Systems Analysis*, 23(6):731–734, 1987.

[39] C. Simone, M. Diehl, M. Jünger, P. Mutzel, G. Reinelt, and G. Rinaldi. Exact ground states of ising spin glasses: new experimental results with a branch-and-cut algorithm. *Journal of Statistical Physics*, 80(1):487–496, 1995.

[40] K. C. Toh, M. J. Todd, and R. H. Tutuncu. SDPT3 — a matlab software package for semidefinite programming. *Optimization Methods and Software*, 11:545–581, 1999.

[41] K. C. Toh, M. J. Todd, and R. H. Tutuncu. Solving semidefinite-quadratic-linear programs using SDPT3. *Mathematical Programming*, 95:189–217, 2003.

[42] G. Weiss. Stochastic bounds on distributions of optimal value functions with applications to pert, network flows and reliability. *Operations Research*, 34:595605, 1986.

Appendix

Proof of Proposition 2. For any $\boldsymbol{c}$ and $\boldsymbol{d} \in \Re^N$, the following holds:

$$\begin{aligned}\max_{\boldsymbol{x}\in\{0,1\}^N}\left(\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x}+\boldsymbol{c}^T\boldsymbol{x}\right) &= \max_{\boldsymbol{x}\in\{0,1\}^N}\left(\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x}+\boldsymbol{d}^T\boldsymbol{x}+(\boldsymbol{c}-\boldsymbol{d})^T\boldsymbol{x}\right)\\ &\leq \max_{\boldsymbol{x}\in\{0,1\}^N}\left(\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x}+\boldsymbol{d}^T\boldsymbol{x}\right)+\sum_{i=1}^N[c_i-d_i]^+.\end{aligned}$$

Taking expectation with respect to the probability measure $P \in \mathbb{P}(P_1, \ldots, P_N)$ and the minimum with respect to $\boldsymbol{d} \in \Re^N$, we obtain

$$\mathbb{E}_P\left[\max_{\boldsymbol{x}\in\{0,1\}^N}\left(\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x}+\boldsymbol{c}^T\boldsymbol{x}\right)\right] \leq \beta^{**}, \quad \forall P \in \mathbb{P}(P_1,\ldots,P_N).$$

Taking supremum with respect to $P \in \mathbb{P}(P_1, \ldots, P_N)$, implies $\beta^* \leq \beta^{**}$.

Next we show $\beta^{**} \leq \beta^*$. Notice that β^{**} can be evaluated as the optimal objective to the following convex programming problem with decision variables $\boldsymbol{d}$ and t:

$$\begin{aligned} \beta^{**} \quad = \quad \min_{\boldsymbol{d},t} \quad & t + \sum_{i=1}^{N} \mathbb{E}_{P_i}[\tilde{c}_i - d_i]^+ \\ \text{s.t.} \quad & t \geq \boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{d}^T\boldsymbol{x}, \qquad \forall \boldsymbol{x} \in \{0,1\}^N. \end{aligned} \tag{5.1}$$

The Karush-Kuhn-Tucker (KKT) conditions for (5.1) are:

$$\lambda(\boldsymbol{x}) \geq 0, t \geq \boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{d}^T\boldsymbol{x}, \quad \forall \boldsymbol{x} \in \{0,1\}^N, \tag{5.2a}$$

$$\sum_{\boldsymbol{x}\in\{0,1\}^N} \lambda(\boldsymbol{x}) = 1, \tag{5.2b}$$

$$\lambda(\boldsymbol{x})(t - \boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} - \boldsymbol{d}^T\boldsymbol{x}) = 0, \quad \forall \boldsymbol{x} \in \{0,1\}^N, \tag{5.2c}$$

$$P(\tilde{c}_i \geq d_i) = \sum_{\boldsymbol{x}\in\{0,1\}^N : x_i = 1} \lambda(\boldsymbol{x}). \tag{5.2d}$$

The Slater's condition for (5.1) is satisfied. Hence there exist dual variables $\lambda(\boldsymbol{x})$ and primal variables $\boldsymbol{d}$, t satisfying the KKT conditions. In the rest of the proof, we let $\boldsymbol{d}, t, \lambda(\boldsymbol{x})$ denote the solutions to the KKT conditions (5.2a)-(5.2d). Let $f_i(\cdot)$ be the probability density function associated with P_i. Next we construct a distribution $\bar{P}$ as follows.

(a) Generate a random vector $\tilde{\boldsymbol{x}}$ which takes the value $\boldsymbol{x} \in \{0,1\}^N$ with probability $\lambda(\boldsymbol{x})$.

(b) Define the set $I_1 = \{i \in [N] : 0 < P(\tilde{c}_i \geq d_i) < 1\}$ and $I_2 = [N] \setminus I_1$. For $i \in I_1$, generate the random variable $\tilde{c}_i$ with the conditional probability density function

$$\bar{f}_i(c_i \mid \tilde{\boldsymbol{x}} = \boldsymbol{x}) = \begin{cases} \dfrac{1}{P(\tilde{c}_i \geq d_i)} \mathbb{I}_{[d_i,\infty)}(c_i) f_i(c_i), & \text{if } x_i = 1, \\ \dfrac{1}{P(\tilde{c}_i < d_i)} \mathbb{I}_{(-\infty,d_i)}(c_i) f_i(c_i), & \text{if } x_i = 0. \end{cases}$$

For $i \in I_2$, generate the random variable $\tilde{c}_i$ with the conditional probability density function $\bar{f}_i(c_i \mid \tilde{\boldsymbol{x}} = \boldsymbol{x}) = f_i(c_i)$.

For $i \in I_1$, the marginal probability density function under $\bar{P}$ is

$$\begin{aligned} \bar{f}_i(c_i) \quad &= \sum_{\boldsymbol{x}\in\{0,1\}^N} \lambda(\boldsymbol{x}) \bar{f}_i(c_i \mid \tilde{\boldsymbol{x}} = \boldsymbol{x}) \\ &= \sum_{\boldsymbol{x}\in\{0,1\}^N : x_i = 1} \lambda(\boldsymbol{x}) \frac{1}{P(\tilde{c}_i \geq d_i)} \mathbb{I}_{[d_i,\infty)}(c_i) f_i(c_i) + \sum_{\boldsymbol{x}\in\{0,1\}^N : x_i = 0} \lambda(\boldsymbol{x}) \frac{1}{P(\tilde{c}_i < d_i)} \mathbb{I}_{(-\infty,d_i)}(c_i) f_i(c_i) \\ &= \mathbb{I}_{[d_i,\infty)}(c_i) f_i(c_i) + \mathbb{I}_{[-\infty,d_i)}(c_i) f_i(c_i) \quad \text{(by (5.2d))} \end{aligned}$$

$$= \quad f_i(c_i).$$

For $i \in I_2$, it is easy to see that $\bar{f}_i(c_i) = f_i(c_i)$. Hence, the constructed probability distribution $\bar{P} \in \mathbb{P}(P_1, \ldots, P_N)$. Therefore

$$\begin{aligned}
\beta^* &\geq \mathbb{E}_{\bar{P}}\left[\max_{\boldsymbol{y}\in\{0,1\}^N} \boldsymbol{y}^T\boldsymbol{Q}\boldsymbol{y} + \tilde{\boldsymbol{c}}^T\boldsymbol{y}\right] \\
&\geq \sum_{\boldsymbol{x}\in\{0,1\}^N} \lambda(\boldsymbol{x})\mathbb{E}_{\bar{P}}\left[\max_{\boldsymbol{y}\in\{0,1\}^N} \boldsymbol{y}^T\boldsymbol{Q}\boldsymbol{y} + \tilde{\boldsymbol{c}}^T\boldsymbol{y} \mid \tilde{\boldsymbol{x}} = \boldsymbol{x}\right] \\
&\geq \sum_{\boldsymbol{x}\in\{0,1\}^N} \lambda(\boldsymbol{x})\mathbb{E}_{\bar{P}}\left[\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \tilde{\boldsymbol{c}}^T\boldsymbol{x} \mid \tilde{\boldsymbol{x}} = \boldsymbol{x}\right] \\
&= \sum_{\boldsymbol{x}\in\{0,1\}^N} \lambda(\boldsymbol{x})\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \sum_{\boldsymbol{x}\in\{0,1\}^N : x_i=1} \lambda(\boldsymbol{x}) \left(\sum_{i\in I_1} \int c_i \frac{1}{P(\tilde{c}_i \geq d_i)} \mathbb{I}_{[d_i,\infty)}(c_i) f_i(c_i) dc_i + \sum_{i\in I_2} \int c_i f_i(c_i) dc_i\right) \\
&= \sum_{\boldsymbol{x}\in\{0,1\}^N} \lambda(\boldsymbol{x})\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \sum_{i\in I_1} \int c_i \mathbb{I}_{[d_i,\infty)}(c_i) f_i(c_i) dc_i + \sum_{i\in I_2} \int c_i P(\tilde{c}_i \geq d_i) f_i(c_i) dc_i.
\end{aligned}$$

Since $P(\tilde{c}_i \geq d_i) = 1$ or 0 for $i \in I_2$, hence $\int c_i P(\tilde{c}_i \geq d_i) f_i(c_i) dc_i = \int c_i \mathbb{I}_{[d_i,\infty)}(c_i) f_i(c_i) dc_i, \forall i \in I_2$. As a result

$$\begin{aligned}
\beta^* &\geq \sum_{\boldsymbol{x}\in\{0,1\}^N} \lambda(\boldsymbol{x})\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \sum_{i=1}^N \int c_i \mathbb{I}_{[d_i,\infty)}(c_i) f_i(c_i) dc_i \\
&= \sum_{\boldsymbol{x}\in\{0,1\}^N} \lambda(\boldsymbol{x})\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \sum_{i=1}^N d_i \int \mathbb{I}_{[d_i,\infty)}(c_i) f_i(c_i) dc_i + \sum_{i=1}^N \int (c_i - d_i)\mathbb{I}_{[d_i,\infty)}(c_i) f_i(c_i) dc_i \\
&= \sum_{\boldsymbol{x}\in\{0,1\}^N} \lambda(\boldsymbol{x})\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \sum_{i=1}^N d_i \sum_{\boldsymbol{x}\in\{0,1\}^N} \lambda(\boldsymbol{x}) x_i + \sum_{i=1}^N \mathbb{E}_{P_i}[\tilde{c}_i - d_i]^+ \qquad \text{(by (5.2}d\text{))} \\
&= \sum_{\boldsymbol{x}\in\{0,1\}^N} \lambda(\boldsymbol{x})(\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{d}^T\boldsymbol{x}) + \sum_{i=1}^N \mathbb{E}_i[\tilde{c}_i - d_i]^+ \\
&= \sum_{\boldsymbol{x}\in\{0,1\}^N} \lambda(\boldsymbol{x}) t + \sum_{i=1}^N \mathbb{E}_i[\tilde{c}_i - d_i]^+ \qquad \text{(by (5.2}c\text{))} \\
&= t + \sum_{i=1}^N \mathbb{E}_i[\tilde{c}_i - d_i]^+ \qquad \text{(by (5.2}b\text{))} \\
&\geq \beta^{**}.
\end{aligned}$$

□

Proof of Proposition 3. First from Proposition 2, we know that

$$\beta^* = \sup_{P_i\in\mathbb{P}_i, i\in[N]} \sup_{P\in\mathbb{P}(P_1,\ldots,P_N)} \mathbb{E}_P\left[\max_{\boldsymbol{x}\in\{0,1\}^N} \boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \tilde{\boldsymbol{c}}^T\boldsymbol{x}\right]$$

$$= \sup_{P_i \in \mathbb{P}_i, i \in [N]} \min_{\boldsymbol{d} \in \Re^N} \Big\{ \max_{\boldsymbol{x} \in \{0,1\}^N} (\boldsymbol{x}^T \boldsymbol{Q} \boldsymbol{x} + \boldsymbol{d}^T \boldsymbol{x}) + \sum_{i=1}^N \mathbb{E}_{P_i} [\tilde{c}_i - d_i]^+ \Big\}.$$

Notice that in the above formula, the objective function is convex with respect to the variable $\boldsymbol{d} \in \Re^N$, and linear with respect to the distribution $P_i \in \mathbb{P}_i, \forall i \in [N]$. Moreover, every probability density function in the distribution set $\mathbb{P}_i$ is bounded in the L_1 space. Hence by Theorem 6 and its corollary in Rockafellar [36], we can exchange the position of sup and min in the above formula. That is

$$\begin{aligned} \beta^* &= \min_{\boldsymbol{d} \in \Re^N} \sup_{P_i \in \mathbb{P}_i, i \in [N]} \Big\{ \max_{\boldsymbol{x} \in \{0,1\}^N} (\boldsymbol{x}^T \boldsymbol{Q} \boldsymbol{x} + \boldsymbol{d}^T \boldsymbol{x}) + \sum_{i=1}^N \mathbb{E}_{P_i} [\tilde{c}_i - d_i]^+ \Big\} \\ &= \min_{\boldsymbol{d} \in \Re^N} \Big\{ \max_{\boldsymbol{x} \in \{0,1\}^N} (\boldsymbol{x}^T \boldsymbol{Q} \boldsymbol{x} + \boldsymbol{d}^T \boldsymbol{x}) + \sum_{i=1}^N \sup_{P_i \in \mathbb{P}_i} \mathbb{E}_{P_i} [\tilde{c}_i - d_i]^+ \Big\} \\ &= \beta^{**}. \end{aligned}$$

□