Hindawi
Journal of Applied Mathematics
Volume 2020, Article ID 5974820, 21 pages
https://doi.org/10.1155/2020/5974820

Research Article

Computational Comparison of Exact Solution Methods for 0-1 Quadratic Programs: Recommendations for Practitioners

Richard J. Forrester and Noah Hunt-Isaak

Dickinson College, Carlisle, Pennsylvania, USA

Correspondence should be addressed to Richard J. Forrester; forrestr@dickinson.edu

Received 28 January 2020; Accepted 12 March 2020; Published 26 April 2020

Academic Editor: Lucas Jodar

This paper is concerned with binary quadratic programs (BQPs), which are among the most well-studied classes of nonlinear integer optimization problems because of their wide variety of applications. While a number of different solution approaches have been proposed for tackling BQPs, practitioners need techniques that are both efficient and easy to implement. We revisit two of the most widely used linearization strategies for BQPs and examine the effectiveness of enhancements to these formulations that have been suggested in the literature. We perform a detailed large-scale computational study over five different classes of BQPs to compare these two linearizations with a more recent linear reformulation and direct submission of the nonlinear integer program to an optimization solver. The goal is to provide practitioners with guidance on how to best approach solving BQPs in an effective and easily implemented manner.

1. Introduction

Binary quadratic programs (BQPs) are one of the most well-studied classes of nonlinear integer optimization problems. These problems appear in a wide variety of applications (see [1, 2] for examples) and are known to be NP-hard. There are a number of different solution techniques that have been proposed for BQPs, including heuristics and exact solution methods. Given the difficulty of implementing and maintaining custom algorithms, the most commonly used exact solution method for BQPs involves linearizing the nonlinear problem and subsequently submitting the equivalent linear form to a standard mixed-integer linear programming (MILP) solver. Interestingly, the majority of commercial MILP solvers have been updated to handle the direct submission of a BQP, which provides an attractive solution alternative.

In this paper, we revisit two of the most widely used linearization strategies for BQPs: the standard linearization [3] and Glover's method [4]. There have been a number of proposed enhancements to these methods, but to the best of our knowledge, there has not been a comprehensive study to determine the optimal manner in which to apply these linearization techniques. In order to investigate these enhancements, we utilize five different classes of BQPs from the literature (the unconstrained BQP, the multidimensional quadratic knapsack problem, the k-item quadratic knapsack problem, the heaviest k-subgraph problem, and the quadratic semi-assignment problem) and three different MILP solvers (CPLEX, GUROBI, and XPRESS). We also make comparisons to the more recent linearization strategy of Sherali and Smith [5] and the direct submission of the BQP to the solver. The goal of this paper is to provide practitioners with suggestions for solving BQPs in an efficient and easily implemented manner. Note that our work is similar to that of [6, 7] who also perform computational studies of different linearization strategies for BQPs. However, our focus is not only on comparing different linearizations but also on how enhancements to the standard linearization and Glover's method affect algorithmic performance.

The general formulation of a BQP is given by

$$\text{BQP}: \text{maximize}\left\{\sum_{i=1}^{n} c_i x_i + \sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n} C_{ij}x_i x_j : x \in \mathbf{X}, x \text{ binary}\right\}, \tag{1}$$

where c_j and C_{ij} are the linear and quadratic cost coefficients, respectively, and X is a polyhedral set representing the feasible region. For notational ease, we henceforth let the indices i and j run from 1 to n unless otherwise stated. Note that for each binary variable, $x_i^2 = x_i$, and thus, C_{ii} can be incorporated into the linear term c_i without loss of generality. Furthermore, note that the overall cost associated with the product $x_i x_j$ is $C_{ij} + C_{ji}$. Therefore, even if C_{ij} and C_{ji} are modified so that their sum is unchanged, the problem remains the same. The two most common quadratic coefficient representations in the literature are to either assume $C_{ij} = C_{ji}$ for all $i \neq j$ (so that the quadratic coefficient matrix C is symmetric) or assume $C_{ij} = 0$ for all $i \geq j$ (so that the quadratic coefficient matrix C is upper triangular); both of which can be assumed without loss of generality. For our purposes, we will explicitly keep both terms $C_{ij}x_i x_j$ and $C_{ji}x_j x_i$. Note that there is no assumption that C is a negative-semidefinite matrix, and therefore, the continuous relaxation of problem BQP is not necessarily a convex optimization problem. However, with regards to the linearizations, the concavity of the objective function is irrelevant as it is rewritten into a linear form.

2. Linearizations

One of the most widely used techniques for optimizing a BQP is to utilize a linearization step that reformulates the nonlinear program into an equivalent linear form through the introduction of auxiliary variables and constraints. The linearized model can then be submitted to a standard MILP solver. While the size and continuous relaxation strength of a reformulation certainly play a role in the performance of a linearization when submitted to an MILP solver, it is not always possible to infer which formulation will have a better performance. Indeed, the use of preprocessing techniques, cutting planes, heuristics, and other enhancements makes it challenging to predict how two linearizations will compare computationally.

In this section, we review two of the most well-known linearization strategies and consider modifications that can affect their computational performance. In addition, we describe the more recent linearization strategy of Sherali and Smith [5].

2.1. Standard Linearization. A standard method to linearize a BQP, due to Glover and Woolsey [3], is to replace each product $x_i x_j$ in the objective function with a continuous variable w_{ij}. In order to simplify the presentation, we first rewrite the objective function of BQP as

$$\sum_{i=1}^{n} c_i x_i + \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} C'_{ij}x_i x_j, \tag{2}$$

where $C'_{ij} = C_{ij} + C_{ji}$ for all (i, j) with $i < j$, which we can do without loss of generality. We now define the *standard* linearization of BQP below.

$$\text{STD}: \text{maximize} \sum_{j=1}^{n} c_j x_j + \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} C'_{ij} w_{ij.} \tag{3}$$

subject to

$$w_{ij} \leq x_i \quad \forall (i, j), i < j, \tag{4}$$

$$w_{ij} \leq x_j \quad \forall (i, j), i < j, \tag{5}$$

$$w_{ij} \geq x_i + x_j - 1 \quad \forall (i, j), i < j, \tag{6}$$

$$w_{ij} \geq 0 \quad \forall (i, j), i < j, \tag{7}$$

$$\mathbf{x} \in \mathbf{X}, \mathbf{x} \text{ binary}. \tag{8}$$

Note that (4)–(7) enforce that $w_{ij} = x_i x_j$ for all binary $\mathbf{x}$. While this linearization method is straightforward, it has the disadvantage that it requires the addition of $n(n-1)/2$ auxiliary variables and $4n(n-1)/2$ auxiliary constraints.

As noted in [8], we can reduce the number of auxiliary constraints by using the sign of C'_{ij}. Let $C^- = \{(i, j): C'_{ij} < 0\}$: and $C^+ = \{(i, j): C'_{ij} > 0\}$. Then, we can omit the constraints (4) and (5) bounding w_{ij} from above for $(i, j) \in C^-$, and we can omit the constraints (6) and (7) bounding w_{ij} from below for $(i, j) \in C^+$, as these will be implied at optimality. We refer to this reduced form as STD$'$ and note that the continuous relaxation of STD$'$ may potentially be weaker than that of STD$'$.

This leads us to the first question that we would like to address:

Question 1: When applying the standard linearization to a BQP, should you reduce the size of the formulation based on the sign of the quadratic objective coefficients? That is, which reformulation, STD or STD$'$, provides a better performance when submitted to a standard MILP solver?

We will address this question in our computational study.

2.2. Glover's Linearization. A more compact linearization method is due to Glover [4]. Given problem BQP, this method replaces each $x_j(\sum_{i=1,i\neq j}^{n} C_{ij}x_i))$ found in the objective function with a continuous variable z_j and uses four linear restrictions to enforce that $z_j = x_j(\sum_{i=1,i\neq j}^{n} C_{ij}x_i)$. Specifically, the *Glover* linearization is as follows.

$$\text{G1}: \text{maximize} \sum_{j=1}^{n} c_j x_j + \sum_{j=1}^{n} z_j \tag{9}$$

subject to

$$z_j \le U_j^1 x_j \quad \forall j, \tag{10}$$

$$z_j \ge L_j^1 x_j \quad \forall i, \tag{11}$$

$$z_j \le \sum_{\substack{i=1 \\ i \ne j}}^{n} C_{ij} x_i - L_j^0 (1 - x_j) \quad \forall j, \tag{12}$$

$$z_j \ge \sum_{\substack{i=1 \\ i \ne j}}^{n} C_{ij} x_i - U_j^0 (1 - x_j) \quad \forall j, \tag{13}$$

$$\mathbf{x} \in \mathbf{X}, \ \mathbf{x} \text{ binary} \tag{14}$$

For each j, U_j^1 and U_j^0 are upper bounds on $\sum_{i=1,i\ne j}^{n} C_{ij} x_i$, while $L_j^1 x_j$ and $L_j^0 x_j$ are lower bounds on $\sum_{i=1,i\ne j}^{n} C_{ij} x_i$. Problems BQP and G1 are equivalent in that, given any binary x, constraints (10)–(13) ensure that $z_j = x_j (\sum_{i=1,i\ne j}^{n} C_{ij} x_i))$ for each j. Observe that this formulation only requires the addition of n (unrestricted) auxiliary continuous variables and $4n$ auxiliary constraints and is therefore considerably more compact than the standard linearization.

As noted earlier, the two most common representations of the quadratic objective coefficients C_{ij} in the literature are to either assume $C_{ij} = C_{ji}$ for all $i < j$ (so that the quadratic coefficient matrix C is symmetric) or assume $C_{ij} = 0$ for all $i \ge j$ (so that the quadratic coefficient matrix C is upper triangular); both of which can be assumed without loss of generality. While the continuous relaxation strength of STD and STD$'$ are not affected by the choice of objective function representation, the relaxation value of G1 is dependent on the manner in which the objective function of a BQP is expressed as shown in [9, 10]. This follows from the fact that the quadratic objective coefficients C_{ij} do not appear in the auxiliary constraints of STD or STD$'$, whereas they *do* appear in those of G1. Moreover, if the quadratic coefficient matrix is upper triangular, then the variable z_1 can be removed from G1 along with the associated constraints in (10)–(13). To see this, note that when C is upper triangular, (10)–(13) ensure that $z_1 = x_1 (\sum_{i=1,i\ne n}^{n} C_{i1} x_i) = 0$ because $C_{ij} = 0$ for all $i \ge j$. Thus, when the quadratic coefficient matrix is upper triangular, Glover's formulation only requires the addition of $n - 1$ auxiliary variables and $4(n - 1)$ auxiliary constraints. Furthermore, when C is upper triangular, the auxiliary constraints of G1 are less dense than when C is symmetric, which may provide a computational advantage when submitted to an MILP solver. This leads us to our next question, which we will address in our computational study.

Question 2: When formulating Glover's formulation, should you represent the quadratic objective coefficient matrix C in upper triangular or symmetric form?

The bounds within G1 can be computed in a number of different ways. As originally suggested in [4], for each j, they can easily be computed as

$$L_j^1 = L_j^0 = \sum_{\substack{i=1, i\ne j \\ C_{ij}<0}}^{n} C_{ij} \text{ and } U_j^1 = U_j^0 = \sum_{\substack{i=1, i\ne j \\ C_{ij}>0}}^{n} C_{ij}. \tag{15}$$

As shown in [9], stronger bounds that take into consideration the feasible region can be computed as

$$L_j^p = \min \left\{ \sum_{\substack{i=1 \\ i\ne j}}^{n} C_{ij} x_i : \mathbf{x} \in \mathbf{X}, x_j = p \right\} \text{ and}$$
$$U_j^p = \max \left\{ \sum_{\substack{i=1 \\ i\ne j}}^{n} C_{ij} x_i : \mathbf{x} \in \mathbf{X}, x_j = p \right\} \tag{16}$$

for $p \in \{0, 1\}$. These bounds could potentially be made even tighter by enforcing x binary so that

$$L_j^p = \min \left\{ \sum_{\substack{i=1 \\ i\ne j}}^{n} C_{ij} x_i : \mathbf{x} \in \mathbf{X}, x_j = p, \mathbf{x} \text{ binary} \right\} \text{ and}$$
$$U_j^p = \max \left\{ \sum_{\substack{i=1 \\ i\ne j}}^{n} C_{ij} x_i : \mathbf{x} \in \mathbf{X}, x_j = p, \mathbf{x} \text{ binary} \right\}. \tag{17}$$

While the continuous relaxation of G1 could be potentially tightened by strengthening the values of the U_j^1, U_j^0, L_j^1, and L_j^0 bounds within the formulation, we need to take into consideration the amount of computational effort required to find the bounds. This leads us to our third question.

Question 3: With regards to the overall computational effort to formulate and solve problem G1 to optimality using an MILP solver, should we compute the bounds U_j^1, U_j^0, L_j^1, and L_j^0 using (15), (16), or (17)?

As noted in [11], we can reduce the size of G1 by removing the constraints (11) and (13) that bound z_j from below because of the maximization objective and the fact that the z_j terms do not appear elsewhere in the problem. Thus,

problem G1 can be written more concisely as the *modified Glover* G2:

$$\text{G2}: \text{maximize} \sum_{j=1}^{n} c_j x_j + \sum_{j=1}^{n} z_j \tag{18}$$

subject to

$$\begin{aligned} &(5), (7) \\ &\mathbf{x} \in \mathbf{X},\ \mathbf{x} \text{ binary}. \end{aligned} \tag{19}$$

Interestingly, problem G2 only requires $2n$ auxiliary constraints as opposed to the $4n$ required for G1, but G2 will have the same continuous relaxation strength as G1.

Our next question to be addressed in the computational study is as follows.

Question 4: How do formulations G1 and G2 compare when submitted to an MILP solver?

It turns out that we can further reduce the number of structural constraints in G2 through the substitution of variables $s_j = U_j^1 x_j - z_j$ or $s_j = \sum_{i \neq j} C_{ij} x_i - L_j^0 (1 - x_j) - z_j$ for all j, which express the variables z_j in terms of the slack variables to the inequalities (10) and (12), respectively (see [11]). Upon performing the first substitution, we obtain G2a:

$$\text{G2a}: \text{maximize} \sum_{j=1}^{n} c_j x_j + \sum_{j=1}^{n} \left(U_j^1 x_j - s_j \right) \tag{20}$$

subject to

$$\begin{aligned} &s_j \geq 0 \quad \forall j, \\ &s_j \geq \left(U_j^1 - L_j^0 \right) x_j - \sum_{\substack{i=1 \\ i \neq j}}^{n} C_{ij} x_i + L_j^0 \quad \forall j, \\ &\mathbf{x} \in \mathbf{X},\ \mathbf{x} \text{ binary}. \end{aligned} \tag{21}$$

The second substitution yields G2b:

$$\text{G2b}: \text{maximize} \sum_{j=1}^{n} c_j x_j + \sum_{j=1}^{n} \left(\sum_{\substack{i=1 \\ i \neq j}}^{n} C_{ij} x_i - L_j^0 (1 - x_j) - s_j \right) \tag{22}$$

subject to

$$\begin{aligned} &s_j \geq \sum_{\substack{i=1 \\ i \neq j}}^{n} C_{ij} x_i - \left(U_j^1 - L_j^0 \right) x_j - L_j^0 \quad \forall j, \\ &s_j \geq 0 \quad \forall j, \\ &\mathbf{x} \in \mathbf{X},\ \mathbf{x} \text{ binary}. \end{aligned} \tag{23}$$

Note that both G2a and G2b only require n (nonnegative) auxiliary variables and n auxiliary constraints, yet they both have the same continuous relaxation strength as G1 and G2. This leads us to our next question:

Question 5: Is it advantageous to perform the substitution of variables to reduce problem G2 to G2a or G2b when submitting the model to an MILP solver? That is, how do formulations G2, G2a, and G2b compare when submitted to an MILP solver?

2.3. Sherali-Smith Linear Formulation. A more recent linearization was introduced by Sherali and Smith [5]. This method converts BQP to the linear form below.

$$\text{SS}: \text{maximize} \sum_{i=1}^{n} s_i + \sum_{i=1}^{n} (c_i + L_i) x_i \tag{24}$$

subject to

$$y_i = \sum_{j=1}^{n} C_{ij} x_j - s_i - L_i \quad \forall i, \tag{25}$$

$$y_i \leq (U_i - L_i)(1 - x_i) \quad \forall i, \tag{26}$$

$$s_i \leq (U_i - L_i) x_i \quad \forall i, \tag{27}$$

$$y_i \geq 0 \quad \forall i, \tag{28}$$

$$s_i \geq 0 \quad \forall i, \tag{29}$$

$$\mathbf{x} \in \mathbf{X},\ \mathbf{x} \text{ binary}. \tag{30}$$

For each i, U_i and L_i are upper and lower bounds, respectively, on $\sum_{j=1, j \neq i}^{n} C_{ij} x_j$, and can be computed in a similar fashion as in (15), (16), or (17). However, based upon preliminary computational results, we will construct SS using the bounds computed as in (16). Note that the authors of [5] actually introduced three formulations, called BP, $\overline{\text{BP}}$, and BP-strong, which all consider instances of BQP that include quadratic constraints. However, each of these three formulations is equivalent to SS for BQP. After performing the substitutions suggested by constraints (25), SS increases the size of the problem by adding an additional n nonnegative auxiliary variables and $3n$ auxiliary constraints (26)–(29).

3. Binary Quadratic Program Classifications

In this section, we introduce the five different classes of BQPs that we consider in our computational study and discuss the manner in which our problem instances are generated.

3.1. Unconstrained 0-1 Quadratic Program: Boolean Least Squares Problem. The first family of 0-1 quadratic problems that we investigate is the unconstrained BQP (UBQP), so that $\mathbf{X}$ is defined as

$$\mathbf{X} \equiv \{\mathbf{x} \in \mathbb{R}^n : 0 \leq x_i \leq 1 \text{ for } i = 1, \cdots, n\}. \tag{31}$$

While simplistic, the UBQP is notable for its ability to represent a wide range of applications (see [2] for a recent survey). For our experiments, we decided to focus on the Boolean least squares problem (BLSP), which is a basic

problem in digital communication where the objective is to identify a binary signal $\mathbf{x}$ from a collection of noisy measurements. Traditionally, this problem is modeled as

$$\text{minimize } \|Dx - \mathbf{d}\|^2 = x^T D^T D x - 2\mathbf{d}^T D x + d^T \mathbf{d} \tag{32}$$

subject to

$$\mathbf{X} \equiv \{\mathbf{x} \in \mathbb{R}^n : 0 \le x_i \le 1 \text{ for } i = 1, \cdots, n\}, \tag{33}$$

where $D \in \mathbb{R}^{m \times n}$ and $\mathbf{d} \in \mathbb{R}^m$ are given. By ignoring the constant term $\mathbf{d}^T\mathbf{d}$, we can rewrite this formulation in our notation and in maximization form by setting $Q = D^T D$, $\mathbf{q} = -2\mathbf{d}^T D$, and subsequently defining $C_{ij} = -Q_{ij}$ for $i \neq j$, $C_{ij} = 0$ for $i = j$, and $c_i = 2q_i + Q_{ii}$ for all i.

We randomly generated instances of BLSP as in [1]. Specifically, we set $m = n$ and generated a random D matrix with elements from the standard normal distribution $N(0, 1)$ and a binary vector $\mathbf{y} \in \mathbb{R}^n$ with elements from the uniform distribution $U(0, 1)$. The vector $\mathbf{d}$ is then constructed as $\mathbf{d} = D\mathbf{y} + \varepsilon$ where ε is a random noise term with elements from $N(0, 1)$. We generated 10 instances for each value of n, which we varied from 30 to 70 in increments of 10 to provide a range of progressively more challenging problems.

3.2. Quadratic Multidimensional Knapsack Problem. The next class of problems that we consider is the quadratic knapsack problem with multiple constraints, also known as the quadratic multidimensional knapsack problem (QMKP) (see [10, 12]). These problems have the following form.

$$\text{QMKP} : \text{maximize} \sum_{j=1}^{n} c_j x_j + \sum_{j=1}^{n} \sum_{\substack{i=1 \\ i \neq j}}^{n} C_{ij} x_i x_j \tag{34}$$

subject to

$$\sum_{i=1}^{n} a_{ki}\, x_i \le b_k \text{ for } k = 1, \cdots m, \qquad \mathbf{x} \text{ binary}. \tag{35}$$

Within QMKP, the c_j, a_{ki}, C_{ij}, and b_k coefficients are typically nonnegative scalars. We assume for every k that $a_{ki} \le b_k$ for each i since otherwise variables can be fixed to 0 and that $b_k < \sum_{i=1}^{n} a_{ki}$ for the kth constraint to be restrictive.

Problem QMKP is a generalization of both the multidimensional knapsack problem and the quadratic knapsack problem (QKP). The multidimensional knapsack problem is that case of QMKP wherein $C_{ij} = 0$ for all (i, j), so that the problem reduces to linear form. The QKP retains the quadratic objective terms but has only a single knapsack constraint defining $\mathbf{X}\,(m = 1)$. This latter problem is one of the most extensively studied nonlinear discrete optimization programs in the literature (see [13] for an excellent survey).

We randomly generated two different test beds; both of which contained instances with $k = 1, 5$ and 10 knapsack constraints. The coefficients a_{ki} are integers taken from a uniform distribution over the interval [1,50], and b_k is an integer from a uniform distribution between 50 and $\sum_{i=1}^{n} a_{ki}$.

For the first test bed, the objective coefficients were all nonnegative. Specifically, the (nonzero) objective coefficients c_j for all j and $C_{ij} = C_{ji}$ for (i, j) with $i < j$ are integers from a uniform distribution over the interval [1,100]. To assess the effects of the density of nonzero c_j and C_{ij} on CPU time, each of these coefficients is nonzero with some predetermined probability Δ. We considered instances with probabilities (densities) $\Delta = 25\%, 50\%, 75\%$, and 100%. For each value of Δ, we generated ten instances for each value of n, which varied from 80 to 110 in increments of 10 when $k = 1$ and varied from 30 to 70 in increments of 10 when $k = 5$ or 10.

For the second test bed, the objective coefficients were mixed in sign. Specifically, the (nonzero) objective coefficients c_j for all j and $C_{ij} = C_{ji}$ for (i, j) with $i < j$ are integers from a uniform distribution over the interval [−100,100]. As with the first test bed, the density of the objective coefficients was varied from $\Delta = 25$–100% in increments of 25%. For each value of Δ, we generated 10 instances for each value of n, which varied from 20 to 50 in increments of 10 for $k = 1, 5$, and 10. We utilized smaller sizes as objective coefficients that are mixed in sign are significantly more difficult to solve.

3.3. The Heaviest k-Subgraph Problem. The heaviest k-subgraph problem (HSP) is concerned with determining a block of k nodes of a weighted graph such that the total edge weight within the subgraph induced by the block is maximized [14]. The HSP can be formulated as

$$\text{HSP} : \text{maximize} \sum_{j=1}^{n} \sum_{\substack{i=1 \\ i \neq j}}^{n} C_{ij} x_i x_j \tag{36}$$

subject to

$$\sum_{i=1}^{n} x_i = k, \qquad \mathbf{x} \text{ binary}. \tag{37}$$

The HSP is also known under the name of the cardinality constrained quadratic binary program, the densest k-subgraph problem, the p-dispersion-sum problem, and the k-cluster problem. For each number of nodes, $n = 10, 20, 30$, and 40, we randomly generated 10 instances with three different graph densities, $\Delta = 25\%, 50\%$, and 75%. Specifically, for each density Δ and any pair of indexes (i, j) such that $i < j$, we assigned $C_{ij} = 1$ with probability Δ and $C_{ij} = 0$ otherwise. We set $k = 0.5\,n$ for all of our tests.

3.4. k-Item Quadratic Knapsack Problem. The k-item quadratic knapsack problem (kQKP), introduced in [15, 16], consists of maximizing a quadratic function subject to a

linear knapsack constraint with an additional equality cardinality constraint:

$$k\text{QKP} : \text{maximize} \sum_{j=1}^{n} c_j x_j + \sum_{j=1}^{n} \sum_{\substack{i=1 \\ i \neq j}}^{n} C_{ij} x_i x_j \tag{38}$$

subject to

$$\sum_{i=1}^{n} a_i x_i \leq b, \tag{39}$$

$$\sum_{i=1}^{n} x_i = k, \tag{40}$$

$$\mathbf{x} \text{ binary}. \tag{41}$$

Here, we assume that $a_i \leq b$ for each i since otherwise variables can be fixed to 0 and that $b < \sum_{i=1}^{n} a_i$ for the constraint to be restrictive. Let us denote by k_{max} the largest number of items which can be filled in the knapsack, that is, the largest number of the smallest a_i whose sum does not exceed b. We can assume that $k \in \{2, \cdots, k_{max}\}$.

Note that kQKP includes two classical subproblems, the HSP by dropping constraint (39) and the QKP by dropping constraint (40). We randomly generated two different tests beds; both of which contained instances with randomly chosen values of k from the discrete uniform distribution over the range $[2, \lfloor n/4 \rfloor]$. For the first test bed, the objective coefficients were all nonnegative, where the (nonzero) objective coefficients c_j for all j and $C_{ij} = C_{ji}$ for (i, j) with $i < j$ are integers from a uniform distribution over the interval [1,100]. The density of the objective coefficients was varied from $\Delta = 25$–100% in increments of 25%. For each value of Δ, we generated ten instances with $n = 50$, 60, and 70 variables. For the second test bed, the objective coefficients were mixed in sign, where the objective coefficients c_j for all j and $C_{ij} = C_{ji}$ for (i, j) with $i < j$ are integers from a uniform distribution over the interval $[-100,100]$. As with the first test bed, the density of the objective coefficients was varied from $\Delta = 25 - 100\%$ in increments of 25%. For each value of Δ, we generated 10 instances for each value of n, where n varied from 50 to 90 in increments of 10.

3.5. Quadratic Semiassignment Problem: Task Allocation Problem. Here, we consider a specific instance of the quadratic semiassignment problem (QSAP) called the task allocation problem [17]. A set of tasks $\{T_1, \cdots, T_m\}$ is to be allocated to a set of processors $\{P_1, \cdots, P_n\}$. The execution costs for assigning task T_i to processor P_k is denoted by e_{ik} and the communication costs between T_i on P_k and T_j on P_l is denoted by C_{ikjl}. The constraints limit every task to exactly one processor. The specific formulation is as follows.

$$\text{QSAP} : \text{minimize} \sum_{i=1}^{m} \sum_{k=1}^{n} e_{ik} x_{ik} + \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} \sum_{k=1}^{n} \sum_{l=1}^{n} C_{ikjl} x_{ik} x_{jl} \tag{42}$$

subject to

$$\sum_{k=1}^{n} x_{ik} = 1 \text{ for } i = 1, \cdots m, \qquad \mathbf{x} \text{ binary}. \tag{43}$$

For our tests, we considered two values of n (4 and 5) and three values of m (10, 12, 15). For each of the six (m, n) pairs, we generated 10 instances where the coefficients e_{ik} and C_{ikjl} are integers from a uniform distribution over the interval $[-50,50]$ as in [1]. Note that in our tests, we changed the sense of the objective function of QSAP to maximization.

4. Computational Study

In this section, we attempt to answer the questions posed in the previous sections. In addition, we compare the standard linearization and Glover's method with both the linearization strategy of Sherali and Smith and direct submission of the BQP to the solver. All tests were implemented in Python and executed on a Dell Precision Workstation equipped with dual 2.4 GHz Xeon processors and 64 GB of RAM running 64bit Windows 10 Professional. In order to better understand the performance of a linear formulation independent of the solver, we utilized three different MILP optimization packages, CPLEX 12.8.0, GUROBI 8.0.0, and XPRESS 8.5.0, which were the most current versions of the software available when we performed the computational tests. We utilized the default settings on all the MILP solvers with a time limit of 3,600 seconds (1 hr) per instance. Note that for the sake of brevity, we only present a subset of the data generated during our study. However, more complete data is available upon request to the corresponding author.

We present our results using *performance profiles*, which is a graphical comparison methodology introduced by Dolan and Moré [18] to benchmark optimization software. Rather than analyzing the data using average solve times that tend to be skewed by a small number of difficult problem instances, performance profiles provide the cumulative distribution function of the ratio of the solve time of a particular formulation versus the best time of all formulations as the performance metric. This effectively eliminates the influence of any outliers and provides a graphical presentation of the expected performance difference among the different formulations.

In order to describe our specific implementation of the performance profiles, suppose that we have a set F of n_f formulations and a set P of n_p problem instances. For each problem p and formulation f, we define $t_{p,f} =$ computing time required to solve problem p using formulation f.

We compare the performance of formulation f on problem p with the best performance by any formulation on this problem using the *performance ratio*

$$r_{p,f} = \frac{t_{p,f}}{\min \{ t_{p,f} : f \in F \}}. \tag{44}$$

Then, the *cumulative performance profile* is defined as

$$\rho_f(\tau) = \frac{1}{n_p} \text{ size } \{p \in P : r_{p,f} \leq \tau\}, \quad (45)$$

so that $\rho_f(\tau)$ is the probability for formulation $f \in F$ that a performance ratio $r_{p,f}$ is within a factor τ of the best possible ratio. Thus, for example, $\rho_f(1)$ is the probability of formulation f to be the best formulation to solve any given problem, while $\rho_f(3)$ is the probability of formulation f to be the best within a time factor of 3 of the best formulation. Note that the performance profiles also provide information about the formulations that fail to reach optimality within the time limit. In particular, those profiles that do not meet $\rho_f(\tau) = 1$ indicate that there is a probability that they will not solve a subset of problems to optimality within the time limit.

4.1. Results for the Standard Linearization. We begin with our first question raised: When implementing the standard linearization, should you utilize the full formulation or use the formulation that is based on the sign of the quadratic objective coefficients? That is, how does STD compare with STD$'$ when submitted to an MILP solver?

Figure 1 shows the performance obtained for STD and STD$'$ for each class of BQP, where we aggregated all of the sizes and densities for each problem class, along with the three different MILP solvers. Note that since STD and STD$'$ vary depending on the sign of the objective coefficients, we provide two performance profiles for both QMKP and k QKP, one for instances with positive quadratic coefficients and one for instances with quadratic coefficients of mixed sign. For the reader unfamiliar with performance profiles, let us examine the results for the BLSP presented in Figure 1(a) in detail. Here, $n_f = 2$ as we are comparing the two formulations STD and STD$'$, while $n_p = 5$ sizes $(n = 30, 40, 50, 60, 70) \times 10$ instances $\times$ 3 MILP solvers = 150. Each of these 150 instances is then solved using STD and STD$'$. For each instance, the minimum computational time of the two formulations is determined, and then the computational time of each formulation is divided by the minimum time, providing the performance ratio. Using the calculation of the performance ratios for all instances, the cumulative profile is then calculated for a set of fixed ratios of time factors. Recall that $\rho_f(1)$ is the probability formulation f will have the fastest solve time for any given problem. Thus, the probability that STD$'$ will have the fastest solve time for BLSP is approximately 86%, whereas the probability that STD will be the fastest formulation is approximately 14%.

The performance profiles clearly indicate that STD$'$ is more likely to have the fastest solve time for all of the BQPs under consideration. While not presented in Figure 1, when aggregated across all problem classes, the probability that STD$'$ is the fastest formulation was 92%. We also note that a more detailed analysis showed that STD$'$ was the superior formulation regardless of objective coefficient density, problem size, or MILP solver—we omit the details for brevity.

While performance profiles provide information on how likely a particular formulation will have a faster solve time, they do not indicate the actual differences in solve time. We report the average CPU times to solve the formulations to proven optimality for the different BQP classes in Figure 2. These average CPU times were computed after removing all instances where at least one of the formulations did not solve within the time limit. That is, the averages only include instances in which both formulations were solved to optimality within the 1-hour time limit. Note that the performance profiles indicate that the number of instances that were not solved within the time limit was fairly small. To see this, recall that those profiles that do not meet $\rho_f(\tau) = 1$ indicate that there is a probability that they will not solve a subset of problems to optimality within the time limit. For example, note that approximately 5% of the instances of HSP formulated with STD did not solve within the 1-hour limit. The data in Figure 2 indicates that STD$'$ is significantly faster than STD on average for all BQP classes considered. Indeed, for 5 of the 7 classifications, the average CPU time of STD$'$ is less than half of that of STD.

Thus, our results clearly indicate that when utilizing the standard linearization, it is best to use the sign-based version STD$'$ regardless of BQP class, objective function density, problem size, or MILP solver. It is interesting to note that STD has a continuous relaxation that is at least as tight as STD$'$ but comes at the expense of a larger problem size. Our tests clearly indicate that the stronger continuous relaxation of STD does not provide an overall computational advantage.

4.2. Results for Glover's Formulation. In this section, we investigate the four questions posed regarding Glover's formulation. While ideally we would investigate these questions simultaneously, doing so would require a prohibitive number of test cases. Therefore, we investigate the questions in an incremental manner to make the task more tractable.

We begin by focusing on answering the second and fourth questions raised. That is, when formulating Glover's model, should we record the quadratic objective coefficient matrix in a symmetric or upper triangular form, and should we reduce the size of G1 by removing constraints (11) and (13) to obtain the modified Glover formulation G2? In order to compare the use of a symmetric quadratic coefficient matrix C with that of one in upper triangular form when formulating G1 and G2, we replaced C_{ij} in the objective function of BQP with C'_{ij} defined as follows. To obtain an upper triangular objective coefficient matrix, we set $C'_{ij} = C_{ij} + C_{ji}$ for all (i, j) with $i < j$ and $C'_{ij} = 0$ for $i \geq j$. To obtain a symmetric objective coefficient matrix, we set $C'_{ij} = C'_{ji} = 1/2(C_{ij} + C_{ji})$ for all (i, j) with $i < j$ and $C'_{ii} = 0$ for all i.

Figure 3 shows the performance of the four different formulations under consideration: G1 and G2 formulated with both symmetric and upper triangular objective coefficient matrices. We once again aggregate across all objective coefficient densities, sizes, and MILP solvers. Note that when

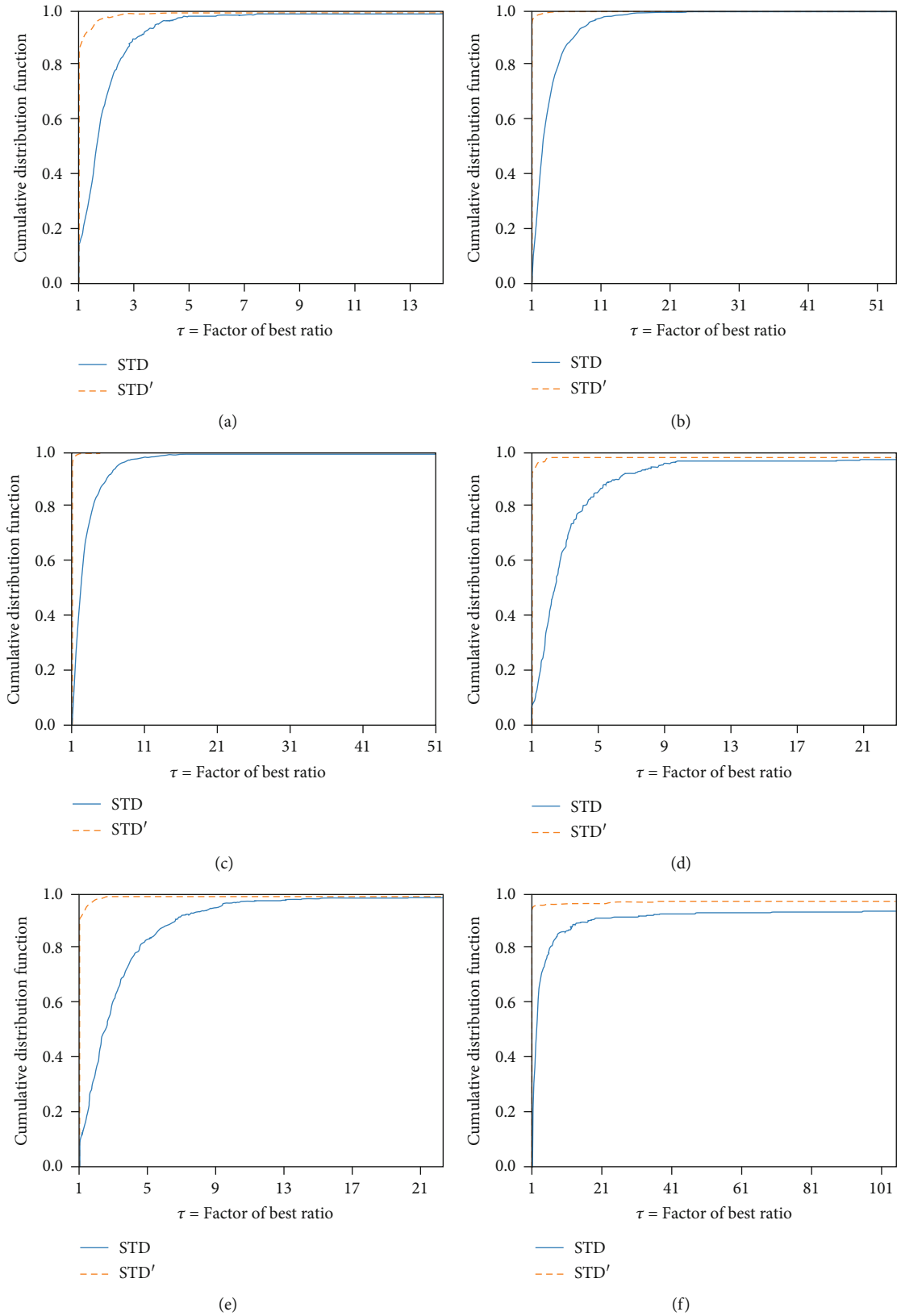
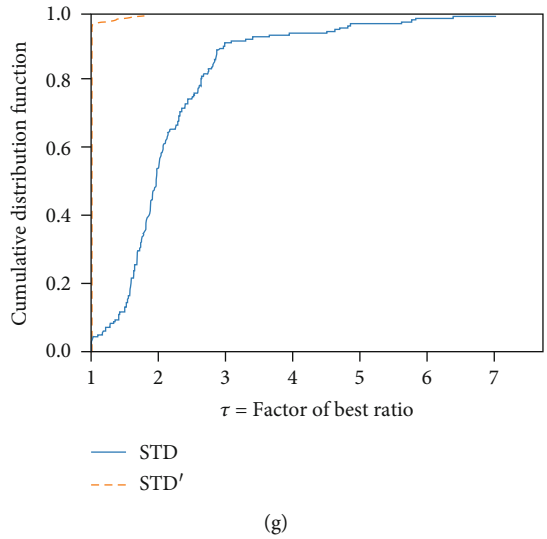


Figure 1: Continued.



(g)

FIGURE 1: Performance profiles based on relative CPU time (s) for Question 1. (a) BLSP, (b) QMKP with positive coefficients, (c) QMKP with mixed-sign coefficients, (d) *k*QKP with positive coefficients, (e) *k*QKP with mixed coefficients, (f) HSP, (g) QSAP.

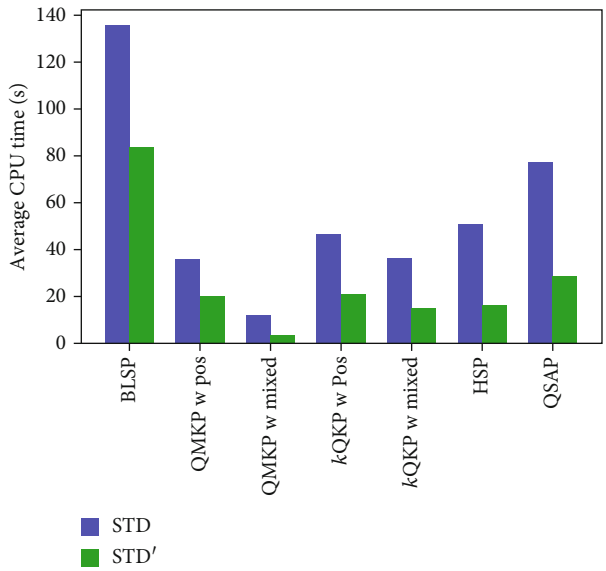


FIGURE 2: Average CPU times (s) for Question 1.

formulating G1 and G2, we computed the bounds within the formulations as in (16) since preliminary computational experience showed this method provides the best performance. However, we will formally compare the bounds of (15), (16), and (17) in another set of tests. We also note that when computing solution times, we included the time needed to find the bounds within G1 and G2.

The results in Figure 3 provide a fairly definitive answer to question 4—Problem G2 is superior to G1 when compared using the same objective coefficient representation. Indeed, G2 with objective coefficients in upper triangular form had the highest probability of being the fastest formulation for all classes of BQPs, while G2 with objective coefficients in symmetric form had the second highest probability of being the fastest formulation for all classes except BLSP and QSAP. With regards to Question 2, note that both G1 and G2 formulated with objective coefficients in upper triangular form were more likely to be faster than their respective versions formulated with symmetric objective coefficients, with the exception of instances of *k*QKP with positive objective coefficients. Interestingly, we note that a more detailed analysis (not presented here) showed that G2 with symmetric objective coefficients was also superior for higher density ($\Delta = 75\%$ and 100%) instances of QMKP with positive coefficients. Note that the performance profiles also indicate that some of the formulations struggled with being able to solve all of the instances within the time limit. For example, G1 and G2 formulated with symmetric objective coefficients were unable to solve approximately 50% of the instances of QSAP, while none of the formulations were able to solve all of the instances of HSP.

We report the average CPU times to solve the formulations to proven optimality for the different BQP classes in Figure 4. As before, these average CPU times were computed after removing all instances where at least one of the formulations did not solve within the time limit.

Figure 4 provides some additional information not captured in the performance profiles. First, note that while the performance profiles in Figure 3 indicate that G2 was superior to G1, Figure 4 indicates that the actual difference in average solution times between G1 and G2 is fairly negligible when they are formulated with the same objective representation. However, there are stark differences between the symmetric and upper triangular coefficient representations for both G1 and G2. In particular, instances of G1 and G2 formulated with symmetric objective coefficients typically take considerably longer to solve on average than their respective models formulated with upper triangular coefficients. There are two exceptions—G2 with symmetric objective coefficients is superior for HSP and instances of *k*QKP with positive coefficients.

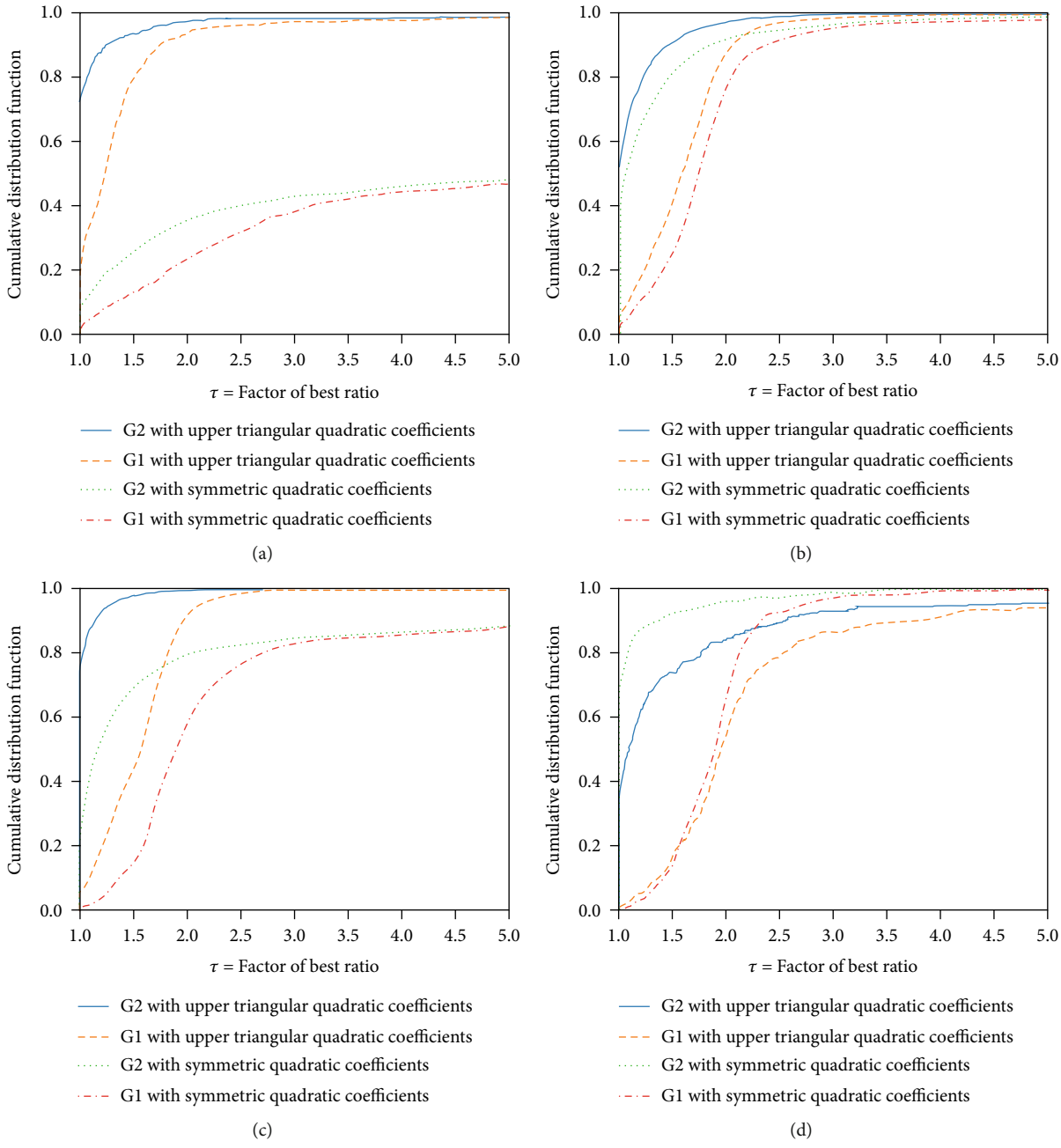


Figure 3: Continued.

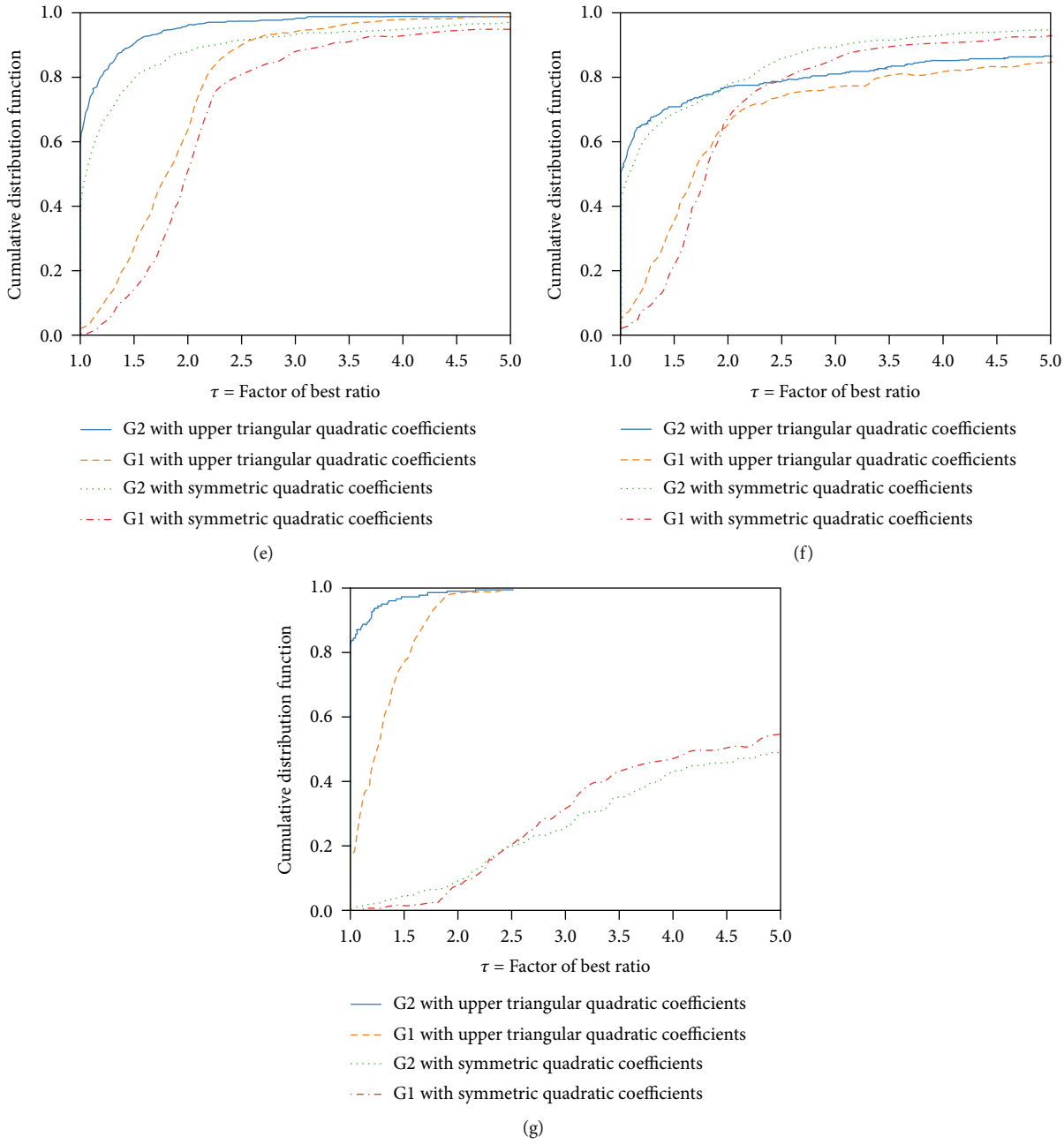


Figure 3: Performance profiles based on relative CPU time (s) for Questions 2 and 4. (a) BLSP, (b) QMKP with positive coefficients, (c) QMKP with mixed-sign coefficients, (d) kQKP with positive coefficients, (e) kQKP with mixed coefficients, (f) HSP, (g) QSAP.

While the results are not completely definitive, Figures 3 and 4 indicate that in general, constraints (11) and (13) should be dropped from G1 to obtain G2 and the quadratic objective coefficients C should be represented in upper triangular form. We reiterate that while the average CPU time differences between G1 and G2 are fairly small, those differences are rather large for the two different objective function representations.

We now attempt to answer Question 5, which is concerned with whether it is advantageous to perform the substitution of variables that reduces G2 to either G2a or G2b. Figure 5 shows the performance profiles of G2, G2a, and G3b where we aggregate across all objective coefficient densities, sizes, and MILP solvers. As with our previous test cases, we computed the bounds within the formulations as in (16).

Note that, typically, G2 was the slowest formulation while G2a and G2b had fairly comparable likelihoods of being the fastest. Although, a closer examination reveals that G2a was slightly more likely to be the fastest formulation for all problem classes except instances of QMKP with positive objective coefficients. We report the average CPU times in Figure 6, which were computed as before.

While the average CPU times for G2, G2a, and G2b are all fairly similar for each problem classification, G2a tended

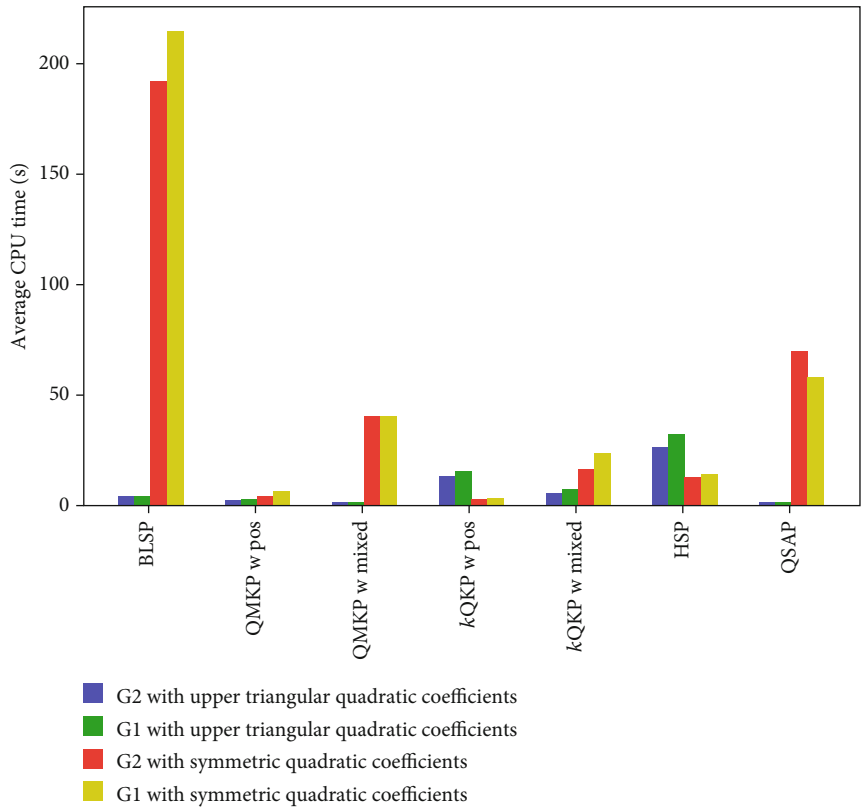


Figure 4: Average CPU times (s) for Questions 2 and 4.

to be the fastest formulation. Therefore, based on our tests, we recommend that G2 should be reduced in size via the substitution of variables $s_j = U_j^1 x_j - z_j$ to obtain G2a. While the results were certainly not definitive, they do suggest that G2a has the highest probability of being the fastest formulation.

We now address Question 3 regarding how the bounds within Glover's formulation should be computed. Within our tests, we refer to the bounds computed as in (15) as *weak*, those computed as in (16) as *tight*, and those as in (17) as *tightest*. Based upon our prior results, our tests were all performed using G2a formulated with the three different methods to compute the bounds. We again reiterate that our solution times include not only the time to solve G2a to integer optimality but also the time required to compute the bounds within the formulation. Our results are presented in Figure 7 where we once again aggregate across all objective coefficient densities, sizes, and MILP solvers. Note that we do not include the BLSP instances because for these problems, the bounds (15), (16), and (17) are all the same.

Interestingly, note that G2a formulated with the weak bounds of (15) tended to be the fastest formulation. We found this result surprising as preliminary experience showed that the tight bounds of (16) were superior. A more detailed analysis revealed that for smaller-sized instances, the extra time needed to find the tighter bounds of (16) was not warranted. However, once the instances start to become more challenging, the additional strength afforded by the tight bounds (16) was advantageous even though the time to find the bounds increases. In order to better understand how these bounds affect CPU time, let us examine the average CPU times presented in Figure 8.

Note that in terms of average CPU times, G2a formulated with the tight bounds (16) was the best formulation for all problem classifications except instances of kQKP with positive objective coefficients. Moreover, not only did the weak bounds (15) perform poorly in terms of average CPU times, they typically performed significantly worse than the tighter bounds. This stems from the fact that the continuous relaxation strength of G2a formulated with (15) for anything other than smaller-sized instances is too weak for the linearization to be effective. Thus, our general recommendation is to formulate Glover's model with the tight bounds of (16), especially for larger-sized instances.

We summarize the answers and recommendations from Questions 1 through 5 in Table 1.

In conclusion, our recommendation based on a detailed computational study across five different classes of BQPs and three different MILP solvers is to implement Glover's formulation using the concise version G2a with an upper triangular objective coefficient matrix C where the bounds within the formulation are computed as in (16).

4.3. Comparison of Exact Solution Methods. In this section, we compare our recommended versions of the standard linearization and Glover's method, STD′, and G2a formulated with an upper triangular quadratic coefficient matrix C and the bounds of (16), with the more recent linearization of

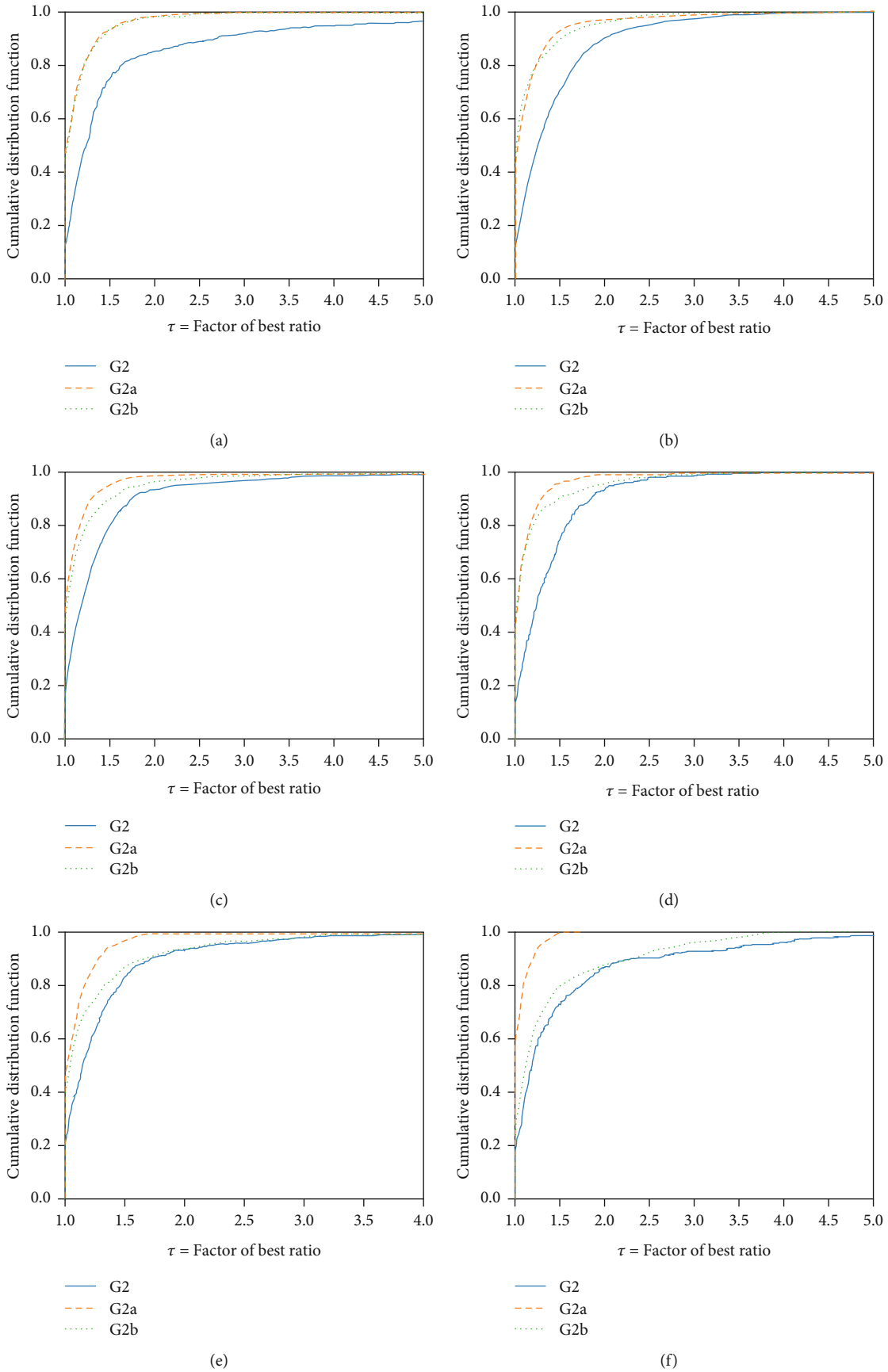
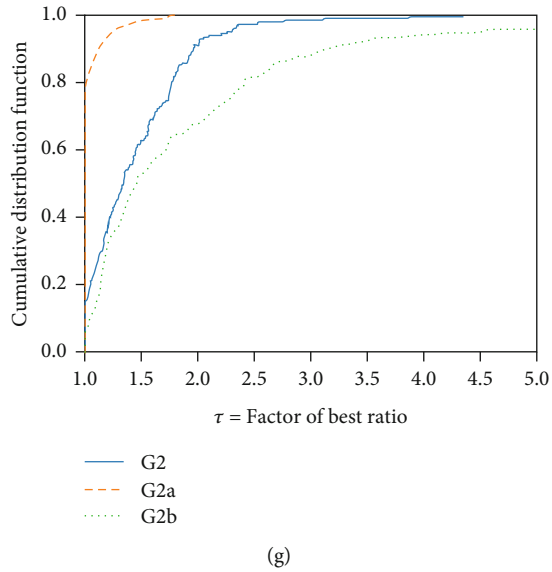


Figure 5: Continued.



(g)

Figure 5: Performance profiles based on relative CPU time (s) for Question 5. (a) BLSP, (b) QMKP with positive coefficients, (c) QMKP with mixed-sign coefficients, (d) *k*QKP with positive coefficients, (e) *k*QKP with mixed coefficients, (f) HSP, (g) QSAP.

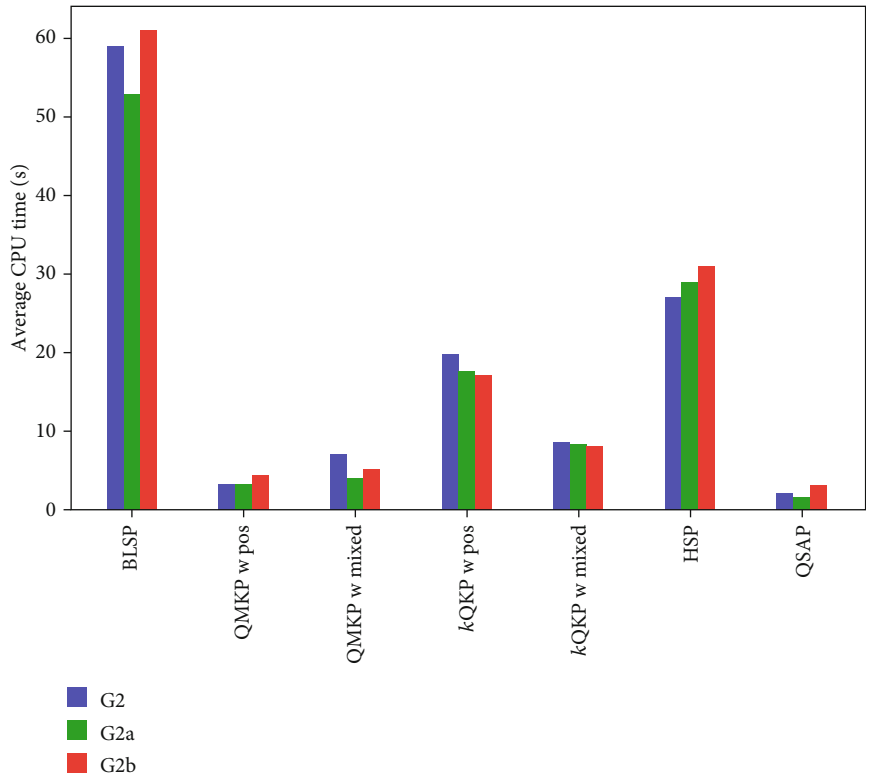


Figure 6: Average CPU times (s) for Question 5.

Sherali and Smith and direct submission to an optimization solver. When formulating the Sherali-Smith model, we utilized the bounds of (16) and an upper triangular quadratic coefficient matrix.

As noted earlier, many MILP solvers have recently been updated to directly handle nonconvex BQPs. Before examining the results of our study, we review the two approaches typically utilized by commercial MILP solvers to solve nonconvex BQPs directly. The MILP solvers of CPLEX, GUROBI, and XPRESS employ one of two different strategies for solving nonconvex BQPs. The first strategy consists of transforming the nonconvex problem into an equivalent convex

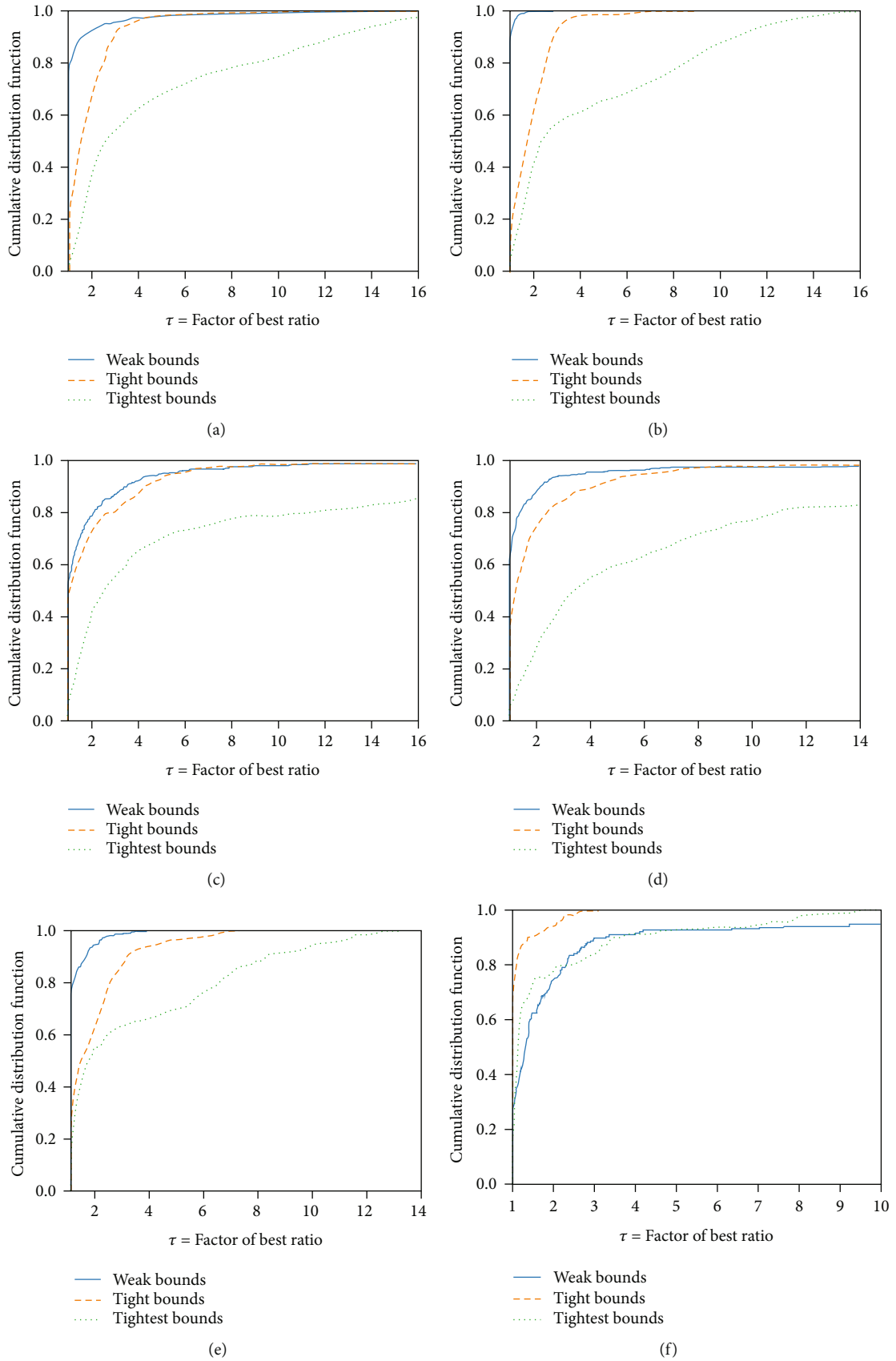


Figure 7: Performance profiles based on relative CPU time (s) for Question 3. (a) QMKP with positive coefficients, (b) QMKP with mixed-sign coefficients, (c) *k*QKP with positive coefficients, (d) *k*QKP with mixed coefficients, (e) HSP, (f) QSAP.

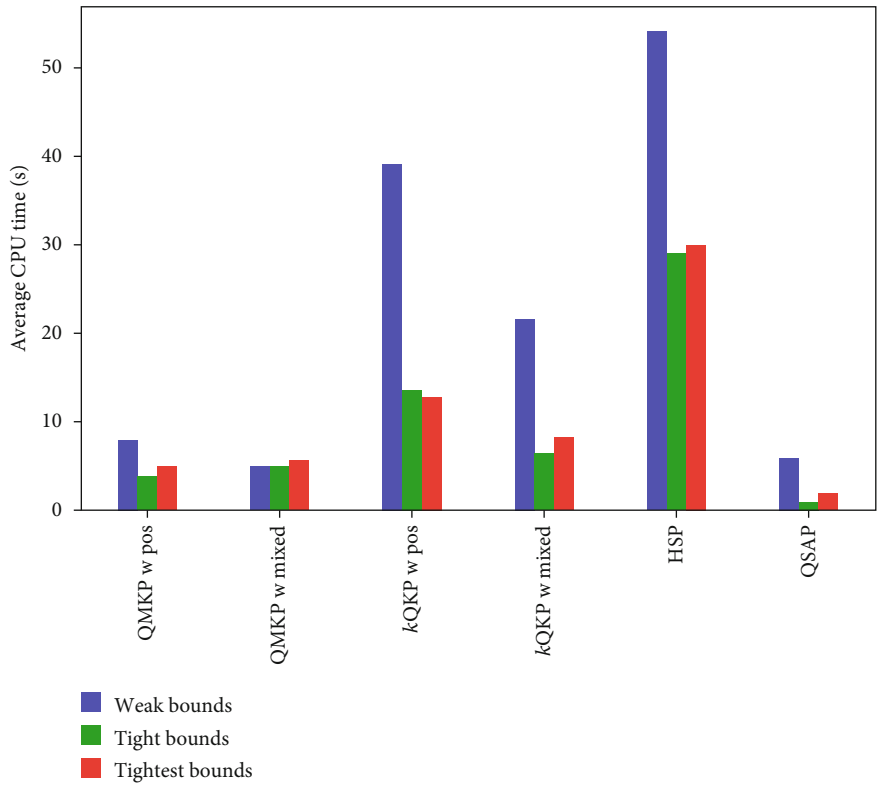


Figure 8: Average CPU times (s) for Question 3.

Table 1: Summary of answers to Questions 1 through 5.

Question	Answer/recommendation
1: When applying the standard linearization to a BQP, should you reduce the size of the formulation based on the sign of the quadratic objective coefficients?	Our recommendation is to use the sign-based formulation STD′ when using the standard linearization.
2: When applying Glover's formulation, should you represent the quadratic objective coefficient matrix C in upper triangular or symmetric form?	Our recommendation is to represent the quadratic objective coefficient matrix C in upper triangular form when using Glover's method.
3: With regards to the overall computational effort to formulate and solve problem G1 to optimality using a MILP solver, should we compute the bounds U_j^1, U_j^0, $L^1{}_j$, and $L^0{}_j$ using (15), (16), or (17)?	In general, we recommend using the bounds of (16) when applying Glover's method. However, for smaller-sized instances it may be beneficial to use the weaker bounds of (15).
4: How do formulations G1 and G2 compare when submitted to a MILP solver?	In general, formulation G2 is superior to G1.
5: Is it advantageous to perform the substitution of variables to reduce Problem G2 to G2a or G2b when submitting the model to a MILP solver?	While formulations G2, G2a, and G2b are fairly comparable in terms of performance, our general recommendation is to use G2a when applying Glover's method.

BQP, which can then be solved using standard convex quadratic programming techniques. This is accomplished by using the identity $\mathbf{x} = \mathbf{x}^T I\mathbf{x}$ when $\mathbf{x}$ is binary. The quadratic portion of the objective function $\mathbf{x}^T C\mathbf{x}$ can therefore be replaced by $\mathbf{x}^T(C + \rho I)\mathbf{x} - \rho\mathbf{x}$. A number of strategies for selecting a ρ so that $C + \rho I$ will be negative semidefinite have been proposed, such as those by Billionet et al. [19]. The second strategy is to employ a linearization technique to transform the nonconvex BQP into an equivalent linear form, such as the standard linearization.

For this second part of our computational study, we randomly generated a completely new set of test problems from those used in Sections 4.1 and 4.2. In addition, we also adjusted the sizes of the problem instances to include more challenging problems (although all other aspects of the instances were generated as presented in Section 3). With regards to instances of Problem QMKP with positive objective coefficients, n was varied from 110 to 140 in increments of 10 when $k = 1$ and varied from 50 to 80 in increments of 10 when $k = 5$ or 10. For instances of QMKP with objective

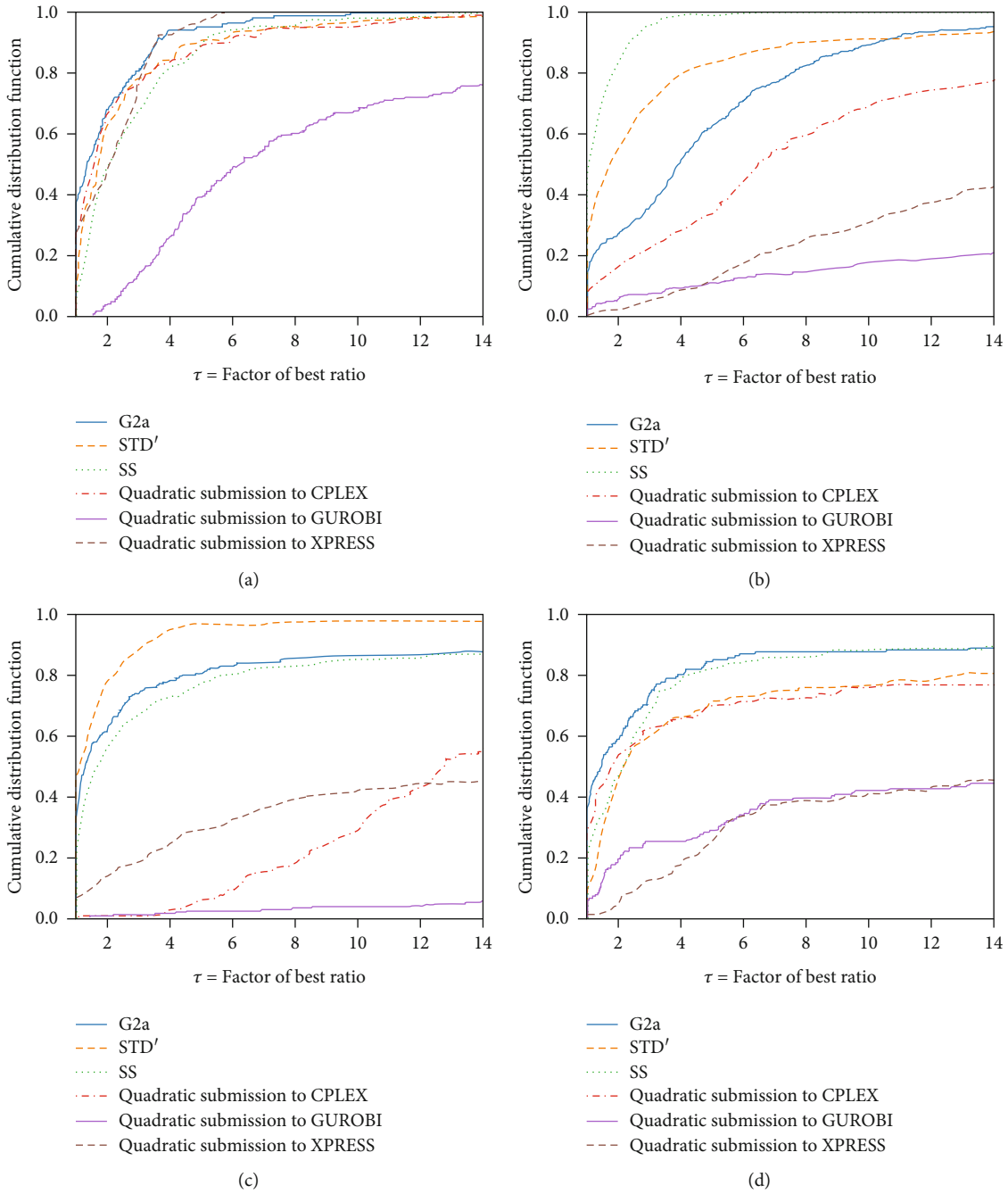


Figure 9: Continued.

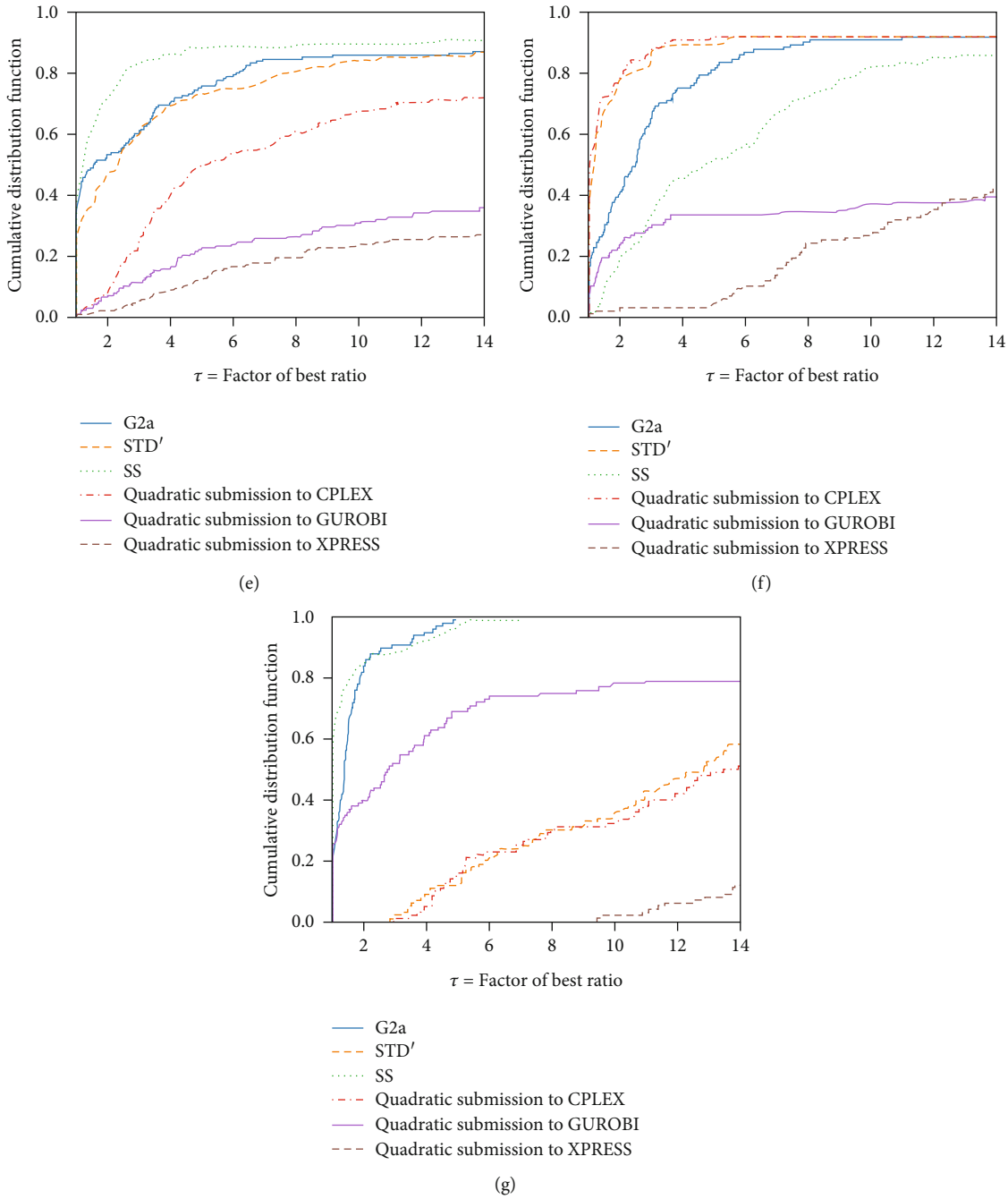


FIGURE 9: Performance profiles based on relative CPU time (s) for all approaches. (a) BLSP, (b) QMKP with positive coefficients, (c) QMKP with mixed-sign coefficients, (d) kQKP with positive coefficients, (e) kQKP with mixed coefficients, (f) HSP, (g) QSAP.

coefficients of mixed sign, n was varied from 40 to 70 in increments of 10 for $k = 1, 5,$ or 10. We varied n from 20 to 50 in increments of 10 for Problem HSP. For instances of kQKP with positive objective coefficients, n was varied from 60 to 90 in increments of 10, while for those instances with objective coefficients of mixed sign, we varied n from 80 to 110 in increments of 10. With regards to Problem QSAP, we considered ten (m, n) pairs where $n \in \{4, 5\}$ and $m \in \{10, 12, 15, 18, 20\}$. For BLSP, we maintained the same sizes as before since this set was already challenging.

The performance profiles of our results are presented in Figure 9, where we once again aggregate across all objective coefficient densities and sizes. Note that in order to ensure that the number of problem instances n_p is the same for each of the formulations as required to construct performance profiles, we only solved the linearizations STD′, G2a, and SS with CPLEX, as opposed to all three solvers as in Sections 4.1 and 4.2. We report the average CPU times in Figure 10, which were computed as before. That is, these average CPU times were computed after removing all instances where at

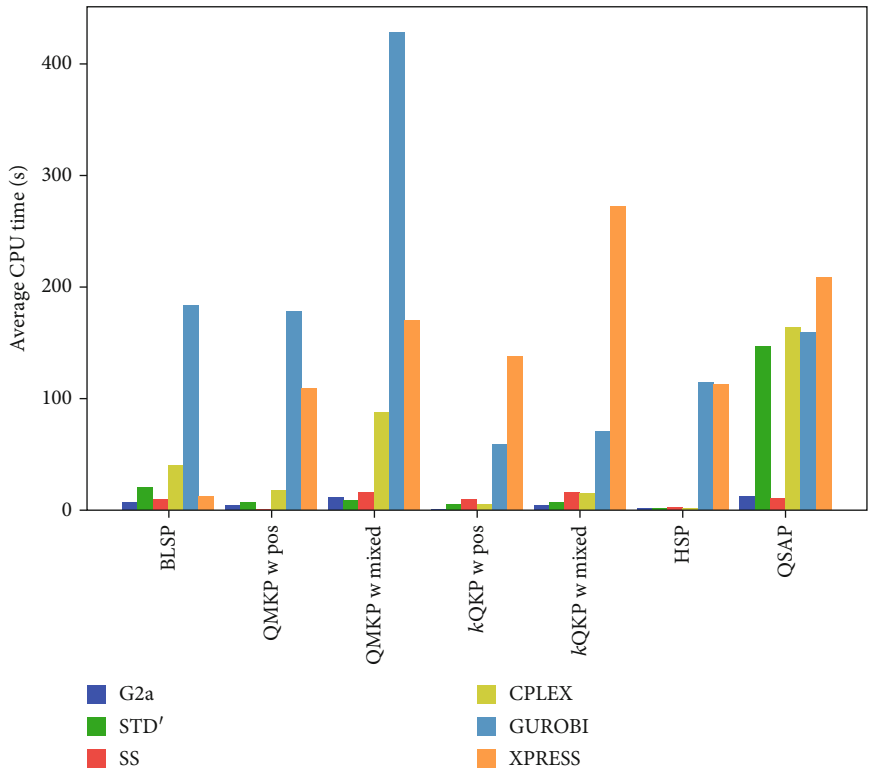


Figure 10: Average CPU times (s) for all approaches.

least one of the formulations did not solve within the time limit. Thus, the averages only include instances in which all formulations were solved to optimality within the 1-hour time limit.

While we had hoped that a single strategy would emerge as the best approach for solving all BQPs regardless of problem class, the results presented in Figures 9 and 10 clearly indicate that this is not the case. Indeed, different strategies appear to be superior for each of the different problem classes. We begin by noting that in general, direct submission of the BQP does not appear to be competitive with submission of any of the linear reformulations. In fact, the average CPU times are typically much higher when using the quadratic solvers of CPLEX, GUROBI, and XPRESS as opposed to solving the linearizations. Moreover, the quadratic solvers were more likely to not be able to solve all instances. For example, the XPRESS solver was unable to solve approximately 55% of the instances of QMKP with objective coefficients of mixed sign. Of the three solvers, CPLEX performed the best for direct submission, which is not surprising given that the nonconvex BQP solver of CPLEX is more mature than those of GUROBI or XPRESS (CPLEX has had the ability to solve nonconvex BQP well before GUROBI or XPRESS). Overall, our tests suggest that while it is convenient to be able to directly submit a BQP to the solver, solution times may be significantly reduced by reformulating the problem into linear form by the user before applying the solver.

We now examine the performance profiles of Figure 9 to compare the linearizations STD′, G2a, and SS. The standard linearization STD′ had the highest probability of being the fastest formulation for instances of QMKP with objective coefficients of mixed sign and instances of HSP. Glover's linearization G2a had the highest probability of being the best formulation to solve instances of BLSP and instances of k QKP with both positive objective coefficients and objective coefficients of mixed sign. The linearization of Sherali and Smith SS had the largest probability of being the fastest formulation for instances of QMKP with positive objective coefficients, instances of kQKP with objective coefficients of mixed sign, and instances of QSAP. When examining the data a little more closely, G2a was the top performing formulation for instances of HSP that were 75% dense, instances of QMKP with objective coefficients of mixed sign that had densities of 75% and 100%, and instances of kQKP with objective coefficients of mixed sign that were 100% dense.

In terms of the average CPU times in Figure 10, there were no significant differences between the three linearizations. All of the average solution times were fairly close with the exception of STD′ applied to instances of QSAP. Indeed, STD′ performed poorly for this class of BQP.

We end our analysis by aggregating the test results for all of the classes of BQPs into a single data set. The performance profiles of the three linearizations are presented in Figure 11, while the average CPU times are presented in Figure 12.

We can see that G2a had the fastest solution time approximately 26% of the time, STD′ was the fastest reformulation approximately 38% of the time, while SS was the superior formulation approximately 34% of the time. Interestingly, in terms of average overall CPU time, we see a different pattern. Note that STD′ required an average overall CPU time of 70

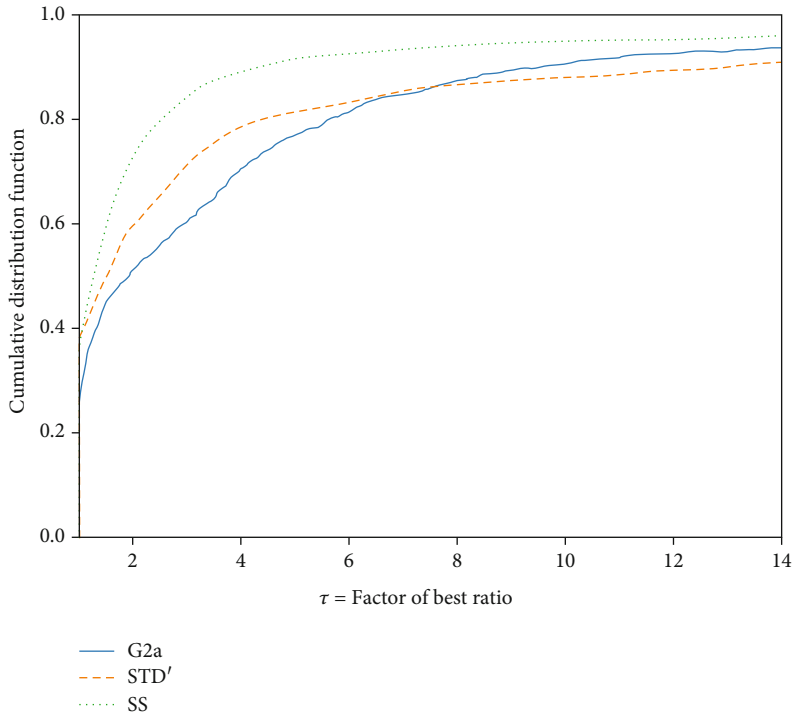


Figure 11: Performance profiles based on relative CPU time (s) over all problem classes.

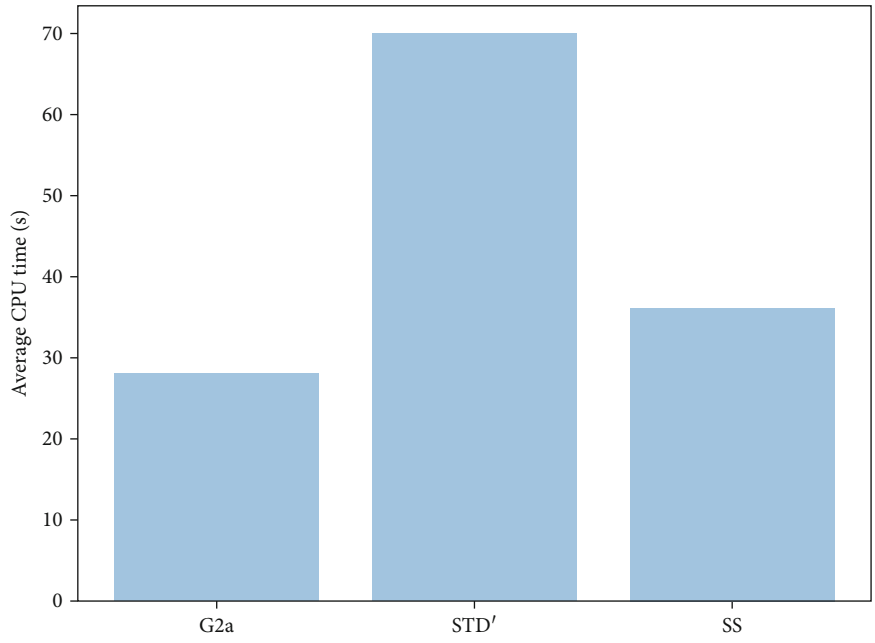


Figure 12: Average CPU times (s) over all problem classes.

seconds, while G2a only required an average of 28 seconds. While we omit the details for brevity, a closer examination of the data suggests that STD′ is better suited for smaller-sized instances of BQPs, likely because of the large number of auxiliary variables and constraints. For larger-sized instances of BQPs, the compactness of G2a and SS seems to provide an advantage when submitted to an MILP solver.

5. Conclusions and Recommendations

In this paper, we revisited the standard linearization [3] and Glover's method [4] for reformulating a BQP into linear form. Using a large-scale computational study over five different classes of BQPs, we evaluated a number of enhancements that have been proposed for these two linearizations.

Based on our analysis, we recommend that when reformulating a BQP using the standard linearization, the sign-based version STD′ should be utilized. When applying Glover's linearization, we recommend using the concise version G2a with an upper triangular objective coefficient matrix C where the bounds within the formulation are computed as in (16).

In the second part of our study, we compared the STD′ and G2a formulations with the more recent linearization of Sherali and Smith and direct submission of the BQP to the solver. Our analysis showed that while it is convenient to submit a BQP directly to the solver, in general, it is advantageous for the user to perform a linear reformulation themselves beforehand. That being said, the more mature nonconvex BQP solver of CPLEX fared acceptably well on a number of different problem classes. Among the linear reformulations studied, no linearization was superior over all problem classes in terms of the probability of having the fastest solution times. In terms of overall average CPU time across all classes of BQPs, G2a and SS yielded the best times. Based upon our analysis, we recommend using STD′ for smaller-sized instances of BQPs, while for larger-sized instances, we recommend using either G2a or SS.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] R. Pörn, O. Nissfolk, A. Skjäl, and T. Westerlund, "Solving 0-1 quadratic programs by reformulation techniques," *Industrial & Engineering Chemistry Research*, vol. 56, no. 45, pp. 13444–13453, 2017.

[2] G. Kochenberger, J. K. Hao, F. Glover et al., "The unconstrained binary quadratic programming problem: a survey," *Journal of Combinatorial Optimization*, vol. 28, no. 1, pp. 58–81, 2014.

[3] F. Glover and E. Woolsey, "Technical Note—Converting the 0-1 polynomial programming problem to a 0-1 linear program," *Operations Research*, vol. 22, no. 1, pp. 180–182, 1974.

[4] F. Glover, "Improved linear integer programming formulations of nonlinear integer Problems," *Management Science*, vol. 22, no. 4, pp. 455–460, 1975.

[5] H. D. Sherali and J. C. Smith, "An improved linearization strategy for zero-one quadratic programming problems," *Optimization Letters*, vol. 1, no. 1, pp. 33–47, 2007.

[6] R. M. Lima and I. E. Grossmann, "On the solution of nonconvex cardinality Boolean quadratic programming problems: a computational study," *Computational Optimization and Applications*, vol. 66, no. 1, pp. 1–37, 2017.

[7] F. Furini and E. Traversi, "Theoretical and computational study of several linearisation techniques for binary quadratic problems," *Annals of Operations Research*, vol. 279, no. 1-2, pp. 387–411, 2019.

[8] R. Forrester and H. Greenberg, "Quadratic binary programming models in computational biology," *Algorithmic Operations Research*, vol. 3, pp. 110–129, 2008.

[9] W. P. Adams, R. J. Forrester, and F. W. Glover, "Comparisons and enhancement strategies for linearizing mixed 0-1 quadratic programs," *Discrete Optimization*, vol. 1, no. 2, pp. 99–120, 2004.

[10] R. J. Forrester, W. P. Adams, and P. T. Hadavas, "Concise RLT forms of binary programs: a computational study of the quadratic knapsack problem," *Naval Research Logistics*, vol. 57, no. 1, pp. 1–12, 2010.

[11] W. P. Adams and R. J. Forrester, "A simple recipe for concise mixed 0-1 linearizations," *Operations Research Letters*, vol. 33, no. 1, pp. 55–61, 2005.

[12] H. Wang, G. Kochenberger, and F. Glover, "A computational study on the quadratic knapsack problem with multiple constraints," *Computers and Operations Research*, vol. 39, no. 1, pp. 3–11, 2012.

[13] D. Pisinger, "The quadratic knapsack problem—a survey," *Discrete Applied Mathematics*, vol. 155, no. 5, pp. 623–648, 2007.

[14] A. Billionnet, "Different formulations for solving the Heaviest *K*-Subgraph problem," *INFOR: Information Systems and Operational Research*, vol. 43, no. 3, pp. 171–186, 2005.

[15] L. Létocart, M. C. Plateau, and G. Plateau, "An efficient hybrid heuristic method for the 0-1 exact *k*-item quadratic knapsack problem," *Pesquisa Operacional*, vol. 34, no. 1, pp. 49–72, 2014.

[16] L. Létocart and A. Wiegele, "Exact solution methods for the *k*-item quadratic knapsack problem," in *International Symposium on Combinatorial Optimization*, pp. 166–176, Springer, Cham, 2016.

[17] 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*, vol. 157, no. 6, pp. 1185–1197, 2009.

[18] E. D. Dolan and J. J. Moré, "Benchmarking optimization software with performance profiles," *Mathematical Programming*, vol. 91, no. 2, pp. 201–213, 2002.

[19] A. Billionnet, S. Elloumi, and A. Lambert, "Extending the QCR method to general mixed-integer programs," *Mathematical Programming*, vol. 131, no. 1-2, pp. 381–401, 2012.