The Quadratic Shortest Path Problem: Complexity, Approximability, and Solution Methods

Borzou Rostami[1], André Chassein[2], Michael Hopf[2], Davide Frey[4], Christoph Buchheim[1], Federico Malucelli[3], Marc Goerigk[5]

[1] Fakultät für Mathematik, TU Dortmund, Germany
[2] Fachbereich Mathematik, TU Kaiserslautern, Germany
[3] Dipartimento di Elettronica e Informazione, Politecnico di Milano, Italy
[4] INRIA-Rennes Bretagne Atlantique, Rennes, France
[5] Department of Management Science, Lancaster University, United Kingdom

Abstract. We consider the problem of finding a shortest path in a directed graph with a quadratic objective function (the QSPP). We show that the QSPP cannot be approximated unless $\mathsf{P} = \mathsf{NP}$. For the case of a convex objective function, an n-approximation algorithm is presented, where n is the number of nodes in the graph, and APX-hardness is shown. Furthermore, we prove that even if only adjacent arcs play a part in the quadratic objective function, the problem still cannot be approximated unless $\mathsf{P} = \mathsf{NP}$. In order to solve the problem we first propose a mixed integer programming formulation, and then devise an efficient exact Branch-and-Bound algorithm for the general QSPP, where lower bounds are computed by considering a reformulation scheme that is solvable through a number of minimum cost flow problems. In our computational experiments we solve to optimality different classes of instances with up to 1000 nodes.

Keywords: Shortest path problem; Quadratic 0–1 optimization; Computational complexity, Branch and Bound.

1 Introduction

The Shortest Path Problem (SPP) of finding a path in a directed graph from an origin node s to a target node t with minimal arc length is a well-studied combinatorial optimization problem. Many classical algorithms such as Dijkstra's labeling algorithm [8] have been developed to solve the SPP efficiently.

Several extensions of the basic SPP exist to model more complex settings. These include problems where the travel costs of an arc follow a distribution and the shortest path is constrained by parameters such as the variance of the cost of the path [20], and problems in which additional costs arise from pairs of arcs in a solution [1].

In this paper we consider the shortest path problem with a quadratic objective function (the QSPP). Specifically, writing the linear objective function of the classical shortest path problem as $c^\top x$ with a cost vector c, the objective function of the QSPP is $x^\top Qx + c^\top x$ with a quadratic matrix Q.

1.1 Applications and Related Work

One variant of the SPP studied in the literature that is directly related to QSPP is that of finding a variance-constrained shortest path [20] where the arc costs are not deterministic but follow a distribution and the objective is to find a path with minimum expected costs subject to the constraint that the variance of the costs is less than a specific threshold. In particular, a solution consists of a path that must have both a short expected length and a low risk of exploding costs in an unfortunate event. An application for this problem is the transportation of hazardous materials. Possible approaches to solve the Variance-Constrained Shortest Path problem involve a relaxation in which the quadratic variance constraint is incorporated into the objective function, thus yielding a QSPP problem. In this case, the quadratic part of the objective function is determined by the covariance matrix of the coefficient's probability distributions, and hence convex. In a similar way, instead of bounding the variance, one may search for a solution that considers both the expected cost and the variance of a path as optimization criteria. [19] consider this as a multi-objective optimization problem. They solve this problem by combining the linear and quadratic objective functions into a single QSPP. Also related to variance-constrained shortest path problems are the so-called reliable shortest paths, see [7].

A different type of applications arises from research on network protocols. [15] study different restoration schemes for self-healing ATM networks. In particular, the authors examine line and end-to-end restoration schemes. In the former, link failures are addressed by routing traffic around the failed link, in the latter, traffic is rerouted by computing an alternative path between source and target. Within their analysis, the authors point out the need to solve a QSPP to address rerouting in the latter scheme. Nevertheless, they do not provide details about the algorithm used to obtain a QSPP solution.

All problems described above involve variants of the classical shortest path problem in which additional costs arise with the presence of pairs of arcs in the solution. Such a setting can be modeled by a quadratic objective function on binary variables associated with each arc, and leads to the definition of a QSPP.

To the best of our knowledge there is no specific method in the literature to solve the QSPP. The only algorithmic approach that has been applied to solve instances of the the QSPP is the one proposed by [4]. They studied a generic framework for solving binary quadratic programming problems. In their computational experiments, they solve some special classes of quadratic $0-1$ problems including the QSPP.

1.2 Main Contributions

In this paper, we analyze the complexity of the general QSPP and several of its special cases. In particular, we show that the general QSPP cannot be ap-

	GRAPH TYPE		
PROBLEM	**general**	**acyclic**	**series-parallel graph**
QSPP	not approximable*	not approximable*	not approximable*
convex QSPP	APX-hard	APX-hard	APX-hard
AQSPP	not approximable*	P	P

Table 1: Our complexity results for different variants of the Quadratic Shortest Path Problem. The entries marked with in asterisk (*) hold true unless NP = P.

proximated unless P = NP. This is done by reducing an instance of the Path with Forbidden Pairs Problem (known to be NP-complete) to a corresponding instance of the QSPP. We also show that, even if we restrict the quadratic part of the cost function to pairs of arcs which are adjacent (AQSPP), the problem still cannot be approximated unless P = NP. This is done by a gap-producing reduction from an instance of 3SAT to an instance of the AQSPP. Moreover, for the convex QSPP where the quadratic form is positive semidefinite and, thus, the objective function is convex, we show that the problem is APX-hard and provide an n-approximation algorithm, where n is number of nodes in the graph. Our complexity results are summarized in Table 1.

From the practical point of view, we present a mixed integer programming formulation whose size is linear in terms of the number of variables in the original quadratic formulation. We also propose an exact Branch-and-Bound algorithm for the general QSPP, where lower bounds are computed by considering a reformulation scheme that is solvable through a number of minimum cost flow problems. In our computational experiments we solve to optimality different types of instances with up to 1000 nodes and show that our results outperform a state-of-the-art solver.

Parts of this paper have been published as conference proceedings [18], where the authors show the NP-hardness of the general QSPP, analyze polynomially solvable special cases, and propose some bounding procedures for the general QSPP.

2 Problem Formulation

Let a directed graph $G(V, A)$ be given, with a source node $s \in V$, a target node $t \in V$, a cost function $c : A \to \mathbb{R}^+$, which maps every arc to a non-negative cost, and a cost function $q : A \times A \to \mathbb{R}^+$ that maps every pair of arcs to a non-negative cost. We denote by $\delta^-(i) = \{j \in V \mid (j,i) \in A\}$ and $\delta^+(i) = \{j \in V \mid (i,j) \in A\}$ the sets of predecessor and successor nodes for any given $i \in V$, by n the number of nodes, and by m the number of arcs. Using binary variables x_{ij} indicating the presence of arc $(i,j) \in A$ on the optimal path, the QSPP is represented as:

$$\begin{aligned} \text{QSPP:} \quad z^* = \min \quad & \sum_{(i,j),(k,l)\in A} q_{ijkl} x_{ij} x_{kl} + \sum_{(i,j)\in A} c_{ij} x_{ij} \\ \text{s.t.} \quad & x \in X_{st},\ x \text{ binary}. \end{aligned} \tag{1}$$

Here the feasible region X_{st} is the path polyhedron

$$X_{st} = \Big\{ 0 \le x \le 1 : \sum_{j\in\delta^+(i)} x_{ij} - \sum_{j\in\delta^-(i)} x_{ji} = b(i) \quad \forall i \in V \Big\}$$

with $b(i) = 1$ for $i = s$, $b(i) = -1$ for $i = t$, and $b(i) = 0$ for $i \in V \setminus \{s,t\}$. Note that, like in the case of classic shortest path problems, it is not necessary to include cycle-elimination constraints, as all costs are positive.

Note that the objective function of the QSPP can be represented by a quadratic and a linear term $f(x) := x^T Qx + c^T x$ for an appropriate matrix Q. We can assume without loss of generality that the matrix Q is symmetric and denote the special case where Q is positive semi-definite, i.e. when f is convex, as the convex QSPP.

Next we define some special cases of the QSPP where the quadratic part of the cost function has a *local* structure, meaning that each pair of variables appearing jointly in a quadratic term in the objective function corresponds to a pair of arcs lying close to each other. We define the Adjacent QSPP (AQSPP), where interaction costs of all non-adjacent pair of arcs are assumed to be zero. Therefore, only the quadratic terms of the form $x_{ij}x_{kl}$ with $j = k$ and $i \neq l$ or with $j \neq k$ and $i = l$ have nonzero objective function coefficients.

As a variant of the AQSPP, we may count additional costs for adjacent arc pairs only if these arcs are traversed consecutively. This problem was investigated in [1, 18]. To distinguish it from the AQSPP, we call it Consecutive QSPP (CQSPP) here. In fact, the AQSPP and the CQSPP are identical if the given graph is acylic. However, for general graphs they are not equivalent. In fact, while the AQSPP is not even approximable in general, as shown in this paper, the CQSPP turns out to be tractable for any graph. This even remains true when taking all arc pairs into account that appear with a fixed maximal distance on the path [18].

3 Complexity Results

3.1 The General QSPP

We start our complexity analysis with the observation that the QSPP can be seen as a generalization of the Path with Forbidden Pairs Problem (PFPP). An instance of the PFPP consists of a graph $G = (V, A)$, two nodes $s, t \in V$ and a list of forbidden arc pairs $\mathcal{L} = \{(a_1, \overline{a}_1), \dots, (a_k, \overline{a}_k)\}$. The goal is to find a path from s to t that contains at most one arc of each arc pair in $\mathcal{L}$. (The problem may also be defined with a list of forbidden vertex pairs). It is known that this problem is NP-complete [9]. Every PFPP can be transformed to an equivalent QSPP, which leads to the following theorem.

Theorem 1. *The QSPP cannot be approximated unless* $\mathsf{P} = \mathsf{NP}$.

Proof. The proof is a reduction from PFPP to QSPP. Given an instance of PFPP, specified by a graph $G = (V, A)$ and a list of forbidden arcs $\mathcal{L}$, we construct an instance of QSPP, specified by a graph G', a cost vector c and a matrix Q. We set $G' := G$ and $c(a) := 0\ \forall a \in A$. Further, we use the quadratic cost function Q of the QSPP to model the forbidden list of arc pairs $\mathcal{L}$. For each arc pair $(a, b) \in \mathcal{L}$, we set $q_{a,b} := 1$. All other entries of Q are zero. Hence, finding a path with costs equal to 0 in G' with respect to the cost function $x^T Qx + c^T x$ is equivalent to finding a path in G that contains at most one arc of each pair in $\mathcal{L}$. Inapproximability follows since a feasible solution of the created QSPP has objective value either 0 or at least 2.

3.2 The Convex QSPP

In the following we consider the convex QSPP. As it turns out it remains APX-hard, but can be approximated within a factor of n. Hence, the non-convexity of the general QSPP is necessary for the non-approximability result of Theorem 1.

Theorem 2. *The convex QSPP is APX-hard.*

Recall that to show that a problem is APX-hard, we have to give a PTAS reduction from another APX-hard problem. For that, we use the Independent Set on degree three graphs problem, which is known to be APX-hard [3].

Independent Set on degree three graphs (IS3)
Given an undirected graph $G = (V, E)$ with node degree at most three for all nodes, find a subset $I' \subset V$ with maximum size such that there exists no edge between two nodes of I'.

Proof. In the following, we construct a PTAS reduction from IS3 to the convex QSPP. A PTAS reduction from a maximization Problem A to a minimization Problem B consists of three polynomial time computable functions $\mathfrak{f}, \mathfrak{g}$, and $\mathfrak{h}$ such that the following relations hold. Let $\mathcal{I}$ be an instance of problem A. Function $\mathfrak{f}$ maps $\mathcal{I}$ to an instance of problem B. The input of $\mathfrak{g}$ is an error parameter ϵ, instance $\mathcal{I}$, and an $(1+\mathfrak{h}(\epsilon))$ approximate solution of the corresponding problem $\mathfrak{f}(\mathcal{I})$. Function $\mathfrak{g}$ produces an solution of $\mathcal{I}$ that is at most $(1-\epsilon)$ times worse than the optimal solution.

In the following, we define the construction that is used to map instances of IS3 to instances of the convex QSPP, i.e., the function $\mathfrak{f}$. Given an instance of IS3 with a graph $G' = (V', E)$ with $V' = \{v_1, \ldots, v_n\}$, we construct the graph $G = (V, A)$ for the instance of the convex QSPP as follows: The node set $V = V' \cup \{v_0\}$ is the node set of the original graph expanded by one additional node v_0. The source node $s = v_0$ and the sink node $t = v_n$. However, the arc set is different

$$A = \{a_i, \overline{a}_i = (v_{i-1}, v_i) | i = 1, \ldots, n\}.$$

We denote in the following all arcs a_i as the top arcs and the arcs $\overline{a}_i$ as the bottom arcs. The graph $G = (V, A)$ is shown in Figure 1.

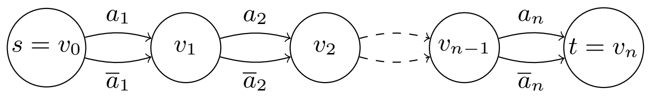

Fig. 1: The graph used for the reduction in the proof of Theorem 2.

Next we give the cost structure that defines the objective function of the convex QSPP. The linear cost vector is set to 0, i.e. $c(a) = 0 \; \forall a \in A$. The costs of the arc pairs are defined as follows:

- $q_{a_i,a_i} = 4 \; \forall i = 1, \ldots, n$
- $q_{\overline{a}_i,\overline{a}_i} = 5 \; \forall i = 1, \ldots, n$
- $q_{a_i,a_j} = 1 \; \forall (i,j)$ with $(v_i, v_j) \in E$

All other arc pairs have zero costs. By construction, the resulting matrix $Q \in \mathbb{R}^{2n\times 2n}$ that represents the quadratic cost term is symmetric. To see that Q is also positive definite, note that, since in G' at most three edges are adjacent to every node, we get that $\sum_{e'\neq e} q_{ee'} \leq 3 \; \forall e \in A$. As $q_{ee} \geq 4$, we can conclude that all eigenvalues of Q must be strictly positive by applying the Gershgorin circle theorem [10].

Next, we describe the function $\mathfrak{g}$. We denote by P an $s-t$ path in G. Every such path contains either a_i or $\overline{a}_i$ for $i = 1, \ldots, n$. Hence, every path P defines a partition of the node set $V' = V_P \cup V_{\bar{P}}$, where $V_P = \{v_i \mid a_i \in P\}$ and $V_{\bar{P}} = \{v_i \mid \overline{a}_i \in P\}$. Given a path P we use this partition to construct an independent set in G' in the following way. If there exists an edge between two nodes of V_P, we remove one of the two nodes. We repeat this deletion procedure until no edge connects two nodes of the set. Denote the so obtained independent set by $\tilde{V_P}$.
The function $\mathfrak{h}$ is defined to be $\mathfrak{h}(\epsilon) = \frac{\epsilon}{19}$.

To show that $\mathfrak{f}, \mathfrak{g}$, and $\mathfrak{h}$ indeed define a PTAS reduction, we have to verify the approximation property. Denote by $f(P)$ the cost of an $s-t$ path P in the convex QSPP instance, by k the size of the maximum independent set $I' \subset V'$ in the original graph G', and by OPT the optimal value of the constructed QSPP instance. We claim that $OPT = 5n - k$. To see that $OPT \leq 5n - k$ consider the following path $\hat{P}$, where arc a_i belongs to $\hat{P}$ if and only if $v_i \in I'$. Then, $f(\hat{P}) = 5(n-k) + 4k = 5n - k$ as I' is an independent set and, hence, no non-diagonal entries of Q must be considered. Assume that $OPT < 5n - k$. Denote by P^* the optimal solution of the convex QSPP instance. We must have that

$|V_{P^*}| > k$ as otherwise a path with cost lower than $5n - k$ is not possible. In this case, however, V_{P^*} cannot be an independent set anymore in G' as the size of the maximum independent set is bounded by k. Therefore, at least one edge must connect two vertices v_j, v_l of V_{P^*}. We can improve the objective value of path P^* by exchanging edge a_j with $\overline{a}_j$ for example. This will decrease the costs of the path, as the diagonal cost of $\overline{a}_j$ is only 1 larger as the diagonal cost of a_j and the costs paid for the two non-diagonal entries will decrease by at least 2, as the number of edges connecting nodes from V_{P^*} is reduced by one and every edge is counted twice. This contradiction shows that $OPT = 5n - k$.

Now let P be a solution of the convex QSPP with $f(P) \leq OPT(1 + \mathfrak{h}(\epsilon))$. The costs of P are given by $f(P) = 5|V_{\bar{P}}| + 4|V_P| + 2|E(V_P)|$, where $E(V_P) \subset E$ are all edges that connect nodes of the set V_P. Using that $|V_{\bar{P}}| = n - |V_P|$ we obtain

$$\begin{aligned} f(P) \leq OPT(1 + \mathfrak{h}(\epsilon)) &\Leftrightarrow 5n - |V_P| + 2|E(V_P)| \leq (5n - k)\left(1 + \frac{\epsilon}{19}\right) \\ &\Leftrightarrow k\left(1 + \frac{\epsilon}{19}\right) - \frac{5n\epsilon}{19} \leq |V_P| - 2|E(V_P)| \end{aligned}$$

The solution that is produced by function $\mathfrak{g}$ is denoted by $\tilde{V_P}$ and we have that $|\tilde{V_P}| \geq |V_P| - |E(V_P)|$, as for every edge connecting two nodes from V_P, at most one node needs to be removed from V_P. The proof is finished if we can show that $|\tilde{V_P}| \geq k(1 - \epsilon)$, as k is the optimal solution value of the original problem. This follows from the following chain of inequalities

$$\begin{aligned} |\tilde{V_P}| &\geq |V_P| - |E(V_P)| \\ &\geq |V_P| - 2|E(V_P)| \\ &\geq k\left(1 + \frac{\epsilon}{19}\right) - \frac{5n\epsilon}{19} \\ &\geq k\left(1 + \frac{\epsilon}{19}\right) - k\frac{20\epsilon}{19} \\ &\geq k(1 - \epsilon), \end{aligned}$$

where we used the fact that $k \geq \frac{n}{4}$ in the penultimate inequality. To get an independent set of this size, just pick an arbitrary node of the vertex set and remove all neighbors of this node from the node set. Note that every node can have at most three neighbors. In this way, at least $\frac{1}{4}$ of all nodes can be picked and no edge will connect two picked nodes.

Theorem 3. *The convex QSPP can be approximated within a factor of n.*

Proof. The objective function of the QSPP is given by the expression $x^TQx + c^Tx$, which can be simplified to $x^T(Q + \text{Diag}\,(c))x$, where $\text{Diag}\,(c)$ is a diagonal matrix with c on the diagonal. This follows from the observation that $x_i = x_i^2 \; \forall x_i \in \{0, 1\}$. Therefore, the objective function of the QSPP can be represented by a single quadratic expression $f(x) := x^TMx$. Without loss of generality we can assume that M is symmetric. Denote by d the diagonal entries of matrix

M. Instead of minimizing function f we can also minimize a function g that approximates f. Consider function $g(x) := x^T \mathrm{Diag}\,(d)x$. We claim that $g(x) \leq f(x) \leq k \cdot g(x)$ for all binary vectors x with k one-entries. As every vector x that represents a simple path has at most n one-entries, we get that $g(x) \leq f(x) \leq n \cdot g(x)$ for all binary vectors representing simple paths. We can restrict the analysis to simple paths as all costs are non negative. Note that it is a classic shortest path problem to solve the problem $\min_{x \in X_{st}} g(x)$, since $x^T \mathrm{Diag}\,(d)x = d^T x$ for all binary x.

We now prove the approximation guarantee of g. As all entries of the matrix M are positive, we have that $g(x) \leq f(x)\ \forall x \geq 0$. The other direction can be seen as follows (without loss of generality we assume that the first k entries of x are one):

$$\begin{aligned}
f(x) = x^T M x &= \sum_{i=1}^{k} M_{ii} x_i^2 + 2 \sum_{i=1}^{k} \sum_{j=i+1}^{k} M_{ij} x_i x_j \\
&= k \sum_{i=1}^{k} M_{ii} x_i^2 - (k-1) \sum_{i=1}^{k} M_{ii} x_i^2 + 2 \sum_{i=1}^{k} \sum_{j=i+1}^{k} M_{ij} x_i x_j \\
&= k \cdot g(x) - (k-1) \sum_{i=1}^{k} M_{ii} x_i^2 + 2 \sum_{i=1}^{k} \sum_{j=i+1}^{k} M_{ij} x_i x_j \\
&= k \cdot g(x) - \sum_{i=1}^{k} \sum_{j=i+1}^{k} M_{ii} x_i^2 - 2 M_{ij} x_i x_j + M_{jj} x_j^2 \\
&\leq k \cdot g(x)
\end{aligned}$$

It remains to show that $0 \leq M_{ii} x_i^2 - 2 M_{ij} x_i x_j + M_{jj} x_j^2$ for all i, j. Define $\hat{x} := e_i \cdot x_i - e_j \cdot x_j$, where e_i is the i^{th} unit vector. As f is a convex function, M must be positive semi definite. Hence, $0 \leq \hat{x}^T M \hat{x} = M_{ii} x_i^2 - 2 M_{ij} x_i x_j + M_{jj} x_j^2$.

Denote by $\tilde{x}$ the path that minimizes g and by x^* the optimal path of the QSPP. Then,

$$f(\tilde{x}) \leq n \cdot g(\tilde{x}) \leq n \cdot g(x^*) \leq n \cdot f(x^*)$$

The first and the last inequality follow from the approximation guarantee of g. The second inequality holds as $\tilde{x}$ is a minimizer of function g.

3.3 The Adjacent QSPP

The next theorem shows that the restriction to the AQSPP does not suffice to reduce the complexity of the problem. A similar (but simpler) reduction can be used to show that the QSPP cannot be approximated unless $\mathsf{P} = \mathsf{NP}$, even if the underlying graphs is series-parallel.

Theorem 4. *The AQSPP cannot be approximated unless* $\mathsf{P} = \mathsf{NP}$.

Proof. We give a gap-producing reduction from 3SAT. Given an instance of 3SAT we create an instance of the AQSPP in polynomial time. If the instance of 3SAT is a yes-instance, i.e., there is an assignment for the literals such that each clause is satisfied, the optimal path of the AQSPP instance has cost zero. Conversely, if the instance of 3SAT is a no-instance, i.e., there is no assignment for the literals such that each clause is satisfied, the optimal path has cost of at least 2. Thus, the existence of an approximation algorithm for AQSPP that runs in polynomial time would imply an algorithm that can decide 3SAT in polynomial time, implying $\mathsf{P} = \mathsf{NP}$.

Let an instance of 3SAT be given in conjunctive normal form containing n literals $x_1, \dots, x_n$ and m clauses $C_1, \dots, C_m$. For convenience, we assume that each clause C_j consists of exactly three literals $x_{j(1)}, x_{j(2)}$, and $x_{j(3)}$ in positive or negative form (the proof also works without this assumption). For the three literals of every clause, there exist 8 possible assignments from which seven satisfy the clause. For example, consider the clause $C_1 = (x_1 \vee \overline{x}_2 \vee x_3)$. The seven satisfying assignments are given by $(x_1, x_2, x_3) = (0,0,0), (0,0,1), (0,1,1), (1,0,0), (1,0,1), (1,1,0)$, or $(1,1,1)$.

Given a 3SAT instance, we construct an instance of AQSPP, specified by a graph $G = (V, A)$, a cost vector c and a matrix Q. The vertex set $V = \{s\} \cup \{v_1, \dots, v_n\} \cup \{C_1, \dots, C_m\} \cup \{t\} \cup V'$ consists of a source node s, one node v_i for each literal x_i, one node C_j for each clause C_j, and a sink node t $(= C_{m+1})$ as well as an additional vertex set V' (cf. Figure 2). The vertex set $V' = \{v_{ijk}, \overline{v}_{ijk} | i \in \{1, \dots, n\}, j \in \{1, \dots, m\}, k \in \{1, \dots, 7\}\}$ consists of $14mn$ vertices that are used to establish an individual connection between each clause and each literal. We connect s and v_1 as well as each v_i and v_{i+1} with two distinctive paths $P_i, \overline{P}_i$ of length $7m + 1$ where

$$P_i = (v_i, v_{i11}, \dots, v_{i17}, v_{i21}, \dots, v_{im7}, v_{i+1}) \text{ and}$$
$$\overline{P}_i = (v_i, \overline{v}_{i11}, \dots, \overline{v}_{i17}, \overline{v}_{i21}, \dots, \overline{v}_{im7}, v_{i+1}).$$

All arcs introduced so far are arcs of type I. Additionally, there is an arc from v_n to the first clause node C_1.

From each clause node C_j, seven paths $Q_{j1}, \dots, Q_{j7}$ are emanating. The arcs of these paths are of type II. Each of these seven paths represents one of the seven feasible assignments of clause C_j. Each of these paths consists of four arcs and connects C_j with C_{j+1}. In the following we give an exact description of path Q_{jk} for clause $C_j = (\tilde{x}_{j(1)} \vee \tilde{x}_{j(2)} \vee \tilde{x}_{j(3)})$. Denote by x' the k^{th} feasible assignment of clause C_j. The first arc points to the node $v_{j(1),j,k}$ if $x'_{j(1)} = 0$, otherwise, it points to the node $\overline{v}_{j(1),j,k}$. The second arc points to the node $v_{j(2),j,k}$ (or $\overline{v}_{j(2),j,k}$) if $x'_{j(2)} = 0$ (or $x'_{j(2)} = 1$). The third arc points to the node $v_{j(3),j,k}$ (or $\overline{v}_{j(3),j,k}$) if $x'_{j(3)} = 0$ (or $x'_{j(3)} = 1$). This might become more clear with a concrete example. Consider again clause $C_1 = (x_1 \vee \overline{x}_2 \vee x_3)$. The first feasible assignment is given by $(x_1, x_2, x_3) = (0,0,0)$, hence the resulting path $Q_{11} = (C_1, v_{1,1,1}, v_{2,1,1}, v_{3,1,1}, C_2)$, the fifth feasible assignment is given by $(x_1, x_2, x_3) = (1,0,1)$, hence, $Q_{15} = (C_1, \overline{v}_{1,1,5}, v_{2,1,5}, \overline{v}_{3,1,5}, C_2)$. Path Q_{11} is shown in Figure 2. Observe that we connect a clause node with the opposite literal assignments.

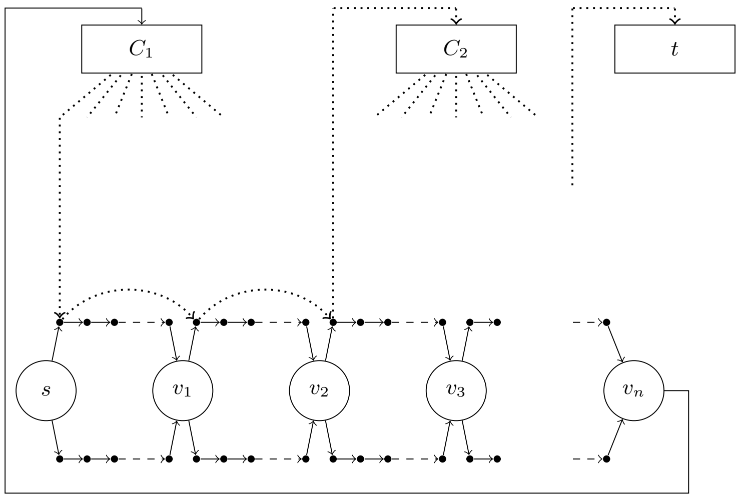

Fig. 2: The graph used for the reduction in the proof of Theorem 4. All horizontal arcs pointing from left to right are of type one, all dotted arcs are of type two. All dashed arcs indicate chains of arcs. The dotted path that is completely shown corresponds to the assignment $x_1 = 0, x_2 = 0$, and $x_3 = 0$ for clause C_1.

Next, we give the description of the cost structure. All linear costs in the corresponding AQSPP instance are zero, i.e., $c(a) = 0$ for all $a \in A$. Quadratic costs occur if and only if two arcs are adjacent and belong to different arc types. All arcs corresponding to the assignments of the clauses, i.e., the arcs on the assignment paths $Q_{j1}, \dots, Q_{j7}$ from C_j to C_{j+1} (as described above) are of type II. All other arcs are of the type I, except of the arc from v_n to C_1.

Next, we show that a 3SAT instance is satisfiable if and only if the optimal solution of the corresponding AQSPP instance has costs zero.

First, suppose the given 3SAT instance is satisfiable. Let x^* be a literal assignment that fulfills all clauses. We need to construct a path P^* in G from s to t with costs zero, thus, without producing quadratic costs. The first part of P^* from s to v_n traverses path P_i if we have $x_i^* = 1$, and $\overline{P}_i$ if we have $x_i^* = 0$. Since x^* is a feasible literal assignment each clause C_j is satisfied. If C_j is satisfied by its k^{th} feasible assignment Q_{jk} is part of P^*. Note that the constructed path is clearly an $s-t$ path. Note further that no quadratic costs can occur since assignment paths Q_{jk} consist only of nodes which correspond to the opposing literal assignment. This may become more clear using a concrete example. Consider the clause $C_1 = (x_1, \overline{x}_2, x_3)$. Assume that this clause is satisfied in the 3SAT instance by the literal assignment $x_1^* = 0, x_2^* = 0$, and $x_3^* = 0$. Hence, $\overline{P}_1, \overline{P}_2$, and $\overline{P}_3$ are part of P^*. As Q_{11} only contains nodes of the paths P_1, P_2, and P_3, no arc pair producing quadratic costs lies on this section of the path (cf. Figure 2).

Conversely, suppose the given 3SAT instance is not satisfiable. We claim that the optimal path of the constructed AQSPP instance has costs of at least 2.

Assume this is not the case and there is a path P' with costs zero. Such a path can never switch from an arc of type I to an arc of type II and vice versa since, then, quadratic costs of at least 2 would occur. Hence, the path P' must traverse from s to v_n, then from C_1 to C_m and finally to t. Thus, the path P' must represent a literal and clause assignment. As the 3SAT instance is a no-instance for each literal assignment, at least one clause, is not satisfied. Let x' be the literal assignment represented by P'. Let C_j be the clause which is not satisfied by x'. Since one of the seven paths $Q_{j1}, \dots, Q_{j7}$ is present in P' and none of the seven feasible assignments of C_j is represented by x', at least one variable is assigned inconsistently. Hence, there exists a node on P' which occurs twice. As the corresponding arcs are of different type quadratic costs of at least 2 occur and we obtain the desired contradiction. Again we use a concrete example to make this more clear. Consider again clause $C_1 = (x_1, \overline{x}_2, x_3)$. Assume that $\overline{P}_1, P_2$, and $\overline{P}_3$ are part of P', i.e. P' represents a literal assignment which does not satisfy clause C_1. Note that $Q_{1k} \cap (\overline{P}_1 \cup P_2 \cup \overline{P}_3) \neq \emptyset$ for $k = 1, \dots, 7$ and, hence, the cost of P' are at least 2.

We conclude the proof with a final remark about the size of the reduction. Graph G consists of $O(mn)$ nodes and arcs. Hence, the reduction is indeed polynomial.

Note that the proof of Theorem 4 can be used to show that the PFPP remains NP-complete even if the list $\mathcal{L}$ is restricted to adjacent arc pairs. The same graph construction is used and the list $\mathcal{L}$ is defined to be all pairs of arcs that have a non zero contribution to the quadratic function. To the best of our knowledge, this result has not been observed yet.

4 Effective Computation of Tight Lower Bounds

Lower bounds are a basic component of Branch-and-Bound algorithms, and a standard tool for the evaluation of heuristic solutions for a minimization problem. In practice, the lack of efficiently computable tight lower bounds can be one of the main reasons for the difficulty of solving even small size instances. However, the choice of the lower bounding procedure should trade off the tightness of the obtained bound and the required computation time. Keeping in mind both the tightness of the bounds and the computational effort to compute these bounds, in this section, we propose lower bounding schemes for the general QSPP based on a closer investigation of the problem structure.

4.1 The Gilmore-Lawler Type Bound

The Gilmore-Lawler (GL) procedure, proposed by [11] and [14], is one of the most popular approaches to find a lower bound for the Quadratic Assignment Problem (QAP) and has been adapted to many other quadratic 0–1 problems in the meantime [5, 16].

For each arc $(i, j) \in A$, potentially in the solution, we consider the minimum interaction cost of (i, j) in a path from s to t. To find these costs we need to compute the shortest among the paths from s to t which contain arc (i,j), using the ij-th row of the quadratic cost matrix as the cost vector. Unfortunately, this problem is NP-complete as it corresponds to the Two Disjoint Paths Problem, which is known to be NP-complete [2]. To avoid computing the exact solution of this problem, we relax the integrality constraints to obtain a minimum cost flow problem. In this way we underestimate the true value of the original problem and, hence, generate also a valid lower bound. Let $\mathcal{P}_{ij}$ be such a subproblem for a given arc $(i,j) \in A$. The minimum cost flow problem contains two origins s and j and two destinations i and t. One unit of flow needs to be transferred from each origin and to each destination. The resulting solution consists either of a path from s to i and from j to t or of the union of a path from s to t that does not contain arc (i,j) and a cycle containing (i,j). These two possibilities are shown in Figure 3

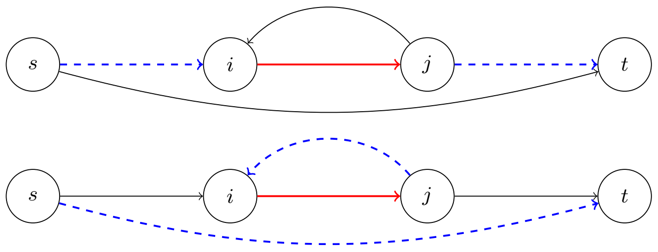

Fig. 3: The two feasible solutions of $\mathcal{P}_{ij}$ are dashed. Note that only the first solution corresponds to an $s-t$ path.

The resulting minimum cost flow problem for each fixed $(i,j) \in A$ is given by:

$$\begin{aligned} \min \quad & \sum_{(k,l)\in A} q_{ijkl}(x_{ij})_{kl} \\ \text{s.t.} \quad & x_{ij} \in X_{st} \\ & (x_{ij})_{ij} = 1 \end{aligned} \qquad (\mathcal{P}_{ij})$$

Denote by z_{ij} the optimal value of $\mathcal{P}_{ij}$. This value underestimates the smallest possible quadratic contribution to the QSPP objective function when arc (i,j) is in the solution. Once z_{ij} has been computed for each $(i,j) \in A$, the GL bound is given by the solution to the following shortest path problem:

$$LB_{GL} = \min \left\{ \sum_{(i,j)\in A} (c_{ij} + z_{ij})x_{ij} : \; x \in X_{st} \right\}.$$

The popularity of the GL approach for computing lower bounds stems from its low computational cost. However, for some quadratic 0–1 problems the obtained bounds deteriorate quickly as the size of the problem increases [6, 17]. To overcome this problem, we present an iterated GL procedure in the following subsection.

4.2 A Reformulation-Based Bound

The GL procedure described above transfers part of the quadratic costs to the linear-cost vector by solving each of the $\mathcal{P}_{ij}$ subproblems. Nevertheless, the part of quadratic costs that is not included in the solutions of $\mathcal{P}_{ij}$ is simply ignored when computing LB_{GL}. Inspired by the reformulation scheme proposed by [6] for the QAP, our next lower bound captures this left-over part by means of the reduced costs associated to the optimal solution of each $\mathcal{P}_{ij}$ subproblem. To define the reduced cost we have to consider the dual problems $\mathcal{D}_{ij}$ of problems $\mathcal{P}_{ij}$. For all $(i,j) \in A$ the dual of $\mathcal{P}_{ij}$ is given by:

$$\begin{aligned} \max\ & (\lambda_{ij})_t - (\lambda_{ij})_s + \pi_{ij} && && (\mathcal{D}_{ij}) \\ \text{s.t.}\ & (\lambda_{ij})_l - (\lambda_{ij})_k \le q_{ijkl} && \forall (k,l) \in A, (k,l) \neq (i,j) \\ & (\lambda_{ij})_j - (\lambda_{ij})_i + \pi_{ij} \le q_{ijij} \end{aligned}$$

For all $(i,j) \in A$ the new linear and quadratic costs are given by

$$\begin{aligned} \tilde{c}_{ij} &= c_{ij} + z_{ij} \\ \tilde{q}_{ijkl} &= q_{ijkl} + (\lambda^*_{ij})_k - (\lambda^*_{ij})_l \quad \forall (k,l) \in A, (k,l) \neq (i,j) \\ \tilde{q}_{ijij} &= q_{ijij} + (\lambda^*_{ij})_i - (\lambda^*_{ij})_j - \pi^*_{ij} \end{aligned}$$

where λ^*_{ij} and π^*_{ij} are the optimal dual values of $\mathcal{D}_{ij}$. Note that the constraints of $\mathcal{D}_{ij}$ ensure that $\tilde{q} \ge 0$. Replacing the costs leads to problem RQSPP, which is equivalent to QSPP, but has increased linear costs.

$$\begin{aligned} \text{RQSPP:} \quad \overline{z}^* = \min\ & \sum_{(i,j),(k,l) \in A} \tilde{q}_{ijkl} x_{ij} x_{kl} + \sum_{(i,j) \in A} \tilde{c}_{ij} x_{ij} \\ \text{s.t.}\ & x \in X_{st},\ x \text{ binary.} \end{aligned} \tag{2}$$

Theorem 5. *Problems QSPP and RQSPP are equivalent.*

Proof. To show that both problems are equivalent, we prove that all feasible solutions $x \in X_{st}$ have the same objective function value. Hence, let $x \in X_{st}$ be arbitrary and fixed. Then

$$\begin{aligned}
&\sum_{(i,j)\in A}\sum_{(k,l)\in A}\tilde{q}_{ijkl}x_{kl}x_{ij}+\sum_{(i,j)\in A}\tilde{c}_{ij}x_{ij}\\
&=\sum_{(i,j)\in A}\left(\sum_{(k,l)\in A}(q_{ijkl}+(\lambda^*_{ij})_k-(\lambda^*_{ij})_l)x_{kl}x_{ij}-\pi^*_{ij}x^2_{ij}\right)+\sum_{(i,j)\in A}(c_{ij}+z_{ij})x_{ij}\\
&=\sum_{(i,j)\in A}\sum_{(k,l)\in A}q_{ijkl}x_{kl}x_{ij}-\sum_{(i,j)\in A}\left(\sum_{(k,l)\in A}((\lambda^*_{ij})_l-(\lambda^*_{ij})_k)x_{kl}x_{ij}+\pi^*_{ij}x_{ij}\right)+\\
&\sum_{(i,j)\in A}c_{ij}x_{ij}+\sum_{(i,j)\in A}z_{ij}x_{ij}\\
&=\sum_{(i,j)\in A}\sum_{(k,l)\in A}q_{ijkl}x_{kl}x_{ij}+\sum_{(i,j)\in A}c_{ij}x_{ij} \qquad (*)
\end{aligned}$$

The last equality $(*)$ can be derived by the following arguments. For all $(i,j)\in A$ we have that

$$\sum_{(k,l)\in A}((\lambda^*_{ij})_l-(\lambda^*_{ij})_k)x_{kl}=(\lambda^*_{ij})_t-(\lambda^*_{ij})_s=z_{ij}-\pi^*_{ij}$$

as x represents an $s-t$ path and strong duality holds between $\mathcal{P}_{ij}$ and $\mathcal{D}_{ij}$. This is equivalent to

$$\sum_{(k,l)\in A}((\lambda^*_{ij})_l-(\lambda^*_{ij})_k)x_{kl}+\pi^*_{ij}=z_{ij}$$

Multiplying with x_{ij} and summing over all $(i,j)\in A$ on both sides yields

$$\sum_{(i,j)\in A}\left(\sum_{(k,l)\in A}((\lambda^*_{ij})_l-(\lambda^*_{ij})_k)x_{kl}x_{ij}+\pi^*_{ij}x_{ij}\right)=\sum_{(i,j)\in A}z_{ij}x_{ij}$$

It can be shown that the procedure, when applied to the reformulated problem again, cannot increase the linear costs anymore, as $z_{ij}=0\ \forall(i,j)\in A$. But, further improvements on the bound may be obtained by directly changing the quadratic cost matrix. Let Π be a matrix with $\Pi_{ij}=-\Pi_{ji}\ \forall i,j$. If $Q+\Pi$ contains only non negative values, the QSPPs with cost matrix Q and $Q+\Pi$ are equivalent, nevertheless, it turns out they may yield to different lower bounds. We propose the following procedure to obtain a strong lower bound: compute the new linear costs $\tilde{c}$ and the new quadratic cost matrix $\tilde{Q}$. Shift the costs of the quadratic cost matrix $\tilde{Q}$ to obtain a symmetric matrix. Repeat this process. This results in a sequence of equivalent QSPP instances $(Q_0,Q_1,\ldots,Q_k$ with $Q_0=$ QSPP), each characterized by a stronger impact of linear costs than the previous ones, and thus providing a better bound. Note that the GLT bound is obtained by considering only the linear portion of the objective function in the first iteration. At some point of this iteration process all subproblems $\mathcal{P}_{ij}$ have an objective value of zero. Hence, no more costs can be transferred from the quadratic to the linear part of the objective function. At this point we stop the process.

4.3 An MILP-Based Bound

In this section we present an MILP formulation for the QSPP which takes advantage of the GL bounds presented in Section 4.1. We associate an overall cost $a_{ij}(x) = c_{ij} + \sum_{(k,l)\in A} q_{ijkl}x_{kl}$ to each arc (i,j) that depends on the arcs that are present in the solution. This allows us to rewrite QSPP as

$$z^* = \min \left\{ \sum_{(i,j)\in A} a_{ij}(x)x_{ij} : \; x \in X_{st} \right\}. \tag{3}$$

If we replace each $a_{ij}(x)$ with its minimum value $c_{ij} + z_{ij}$ over the set of possible feasible solutions where arc (i,j) is in the solution, the GL bound is obtained. Let us define a new variable $y_{ij} = a_{ij}(x)x_{ij}$ for all $(i,j) \in A$. Therefore, we have

$$y_{ij} \geq (c_{ij} + z_{ij})x_{ij} \quad (i,j) \in A. \tag{4}$$

Moreover, let w_{ij} represent an upper bound on the cost $\sum_{(k,l)\in A} q_{ijkl}x_{kl}$. In principle, we can compute w_{ij} by setting $w_{ij} = \sum_{(k,l)\in A} q_{ijkl}$. However, taking into account the structure of the graph, a better estimation may be obtained. For acyclic graphs, for example, w_{ij} can be computed by solving the following minimum cost flow problem:

$$w_{ij} = \max\{ \sum_{(k,l)\in A} q_{ijkl}x_{kl} : x \in X_{st}\} = -\min\{ \sum_{(k,l)\in A} -q_{ijkl}x_{kl} : x \in X_{st}\}.$$

Following the well-known results of [12], we can derive the following inequality:

$$y_{ij} \geq \sum_{(k,l)\in A} q_{ijkl}x_{kl} - w_{ij}(1 - x_{ij}) + c_{ij}x_{ij} \quad (i,j) \in A. \tag{5}$$

Using (4) and (5) the QSPP can be linearized as follows:

$$\begin{aligned}
\text{MILP:} \quad z^* = \min \quad & \sum_{(i,j)\in A} y_{ij} \\
\text{s.t. } & y_{ij} \geq (c_{ij} + z_{ij})x_{ij} && (i,j) \in A \\
& y_{ij} \geq \sum_{(k,l)\in A} q_{ijkl}x_{kl} - w_{ij}(1 - x_{ij}) + c_{ij}x_{ij} && (i,j) \in A \\
& x \in X_{st}, \; x \text{ binary.}
\end{aligned}$$

Observe that an optimal solution to the MILP will yield an optimal solution to the QSPP. However, if the binary restrictions on variables x are relaxed in the MILP, the problem is no longer equivalent to the QSPP, providing a lower bound on the optimal value of the QSPP.

5 The Branch-and-Bound Algorithm

In this section we describe our approach to incorporating the previous lower bounds into a Branch-and-Bound strategy in order to obtain an optimal solution of the QSPP. More specifically, The application of Branch-and-Bound to the QSPP requires a method to obtain a lower bound, a method to obtain a feasible solution (and an upper bound), and a method to partition the feasible region of a given problem (branching rules). The first two requirements are automatically satisfied by any of the lower bounds we described in Sections 4.1 and 4.2 as their application also provides feasible QSPP solutions.

At the root node of the branching tree, we apply the reformulation-based lower bound to define tight upper and lower bounds followed by a reduction procedure in which we try to fix the values of some variables. The nodes of the branching tree other than the root node are processed quite fast, without the reformulation procedure. We simply solve a linear SPP with the linear costs found by the reformulation scheme at the root node, possibly updating the incumbent solution and applying branching. We should note here that we also considered different versions of the branch-and-bound algorithm with different combinations of using the GL and reformulation procedures at the root node and at the other nodes of the branching tree. However, the overall computing times were much worse in these cases.

To address the branching strategy we need to consider the nature of the problem in an efficient way. This is done in the following subsection.

5.1 Branching Strategy

Given a source node s and a target node t, a feasible solution to our problem is a path connecting these two nodes. A simple way to partition the solution space is therefore to consider the subproblems associated with each of the neighbors of the start node. The solution to each of these subproblem consists of the combination of the arc from the start node to the neighbor v, and the solution to a quadratic shortest path problem from v to the target node. This idea must clearly be refined to obtain a correct subproblem when forcing a neighbor as a new start node. Let us consider what happens when we partition the original problem by considering the paths that start with one of the neighbors of the start node. Let s' be the current start node and v be the considered neighbor. Our branching step involves forcing arc (s', v) into the solution. This means that the cost of the paths starting from v will have to include the cost of arc (s', v). The quadratic contribution of arc (s', v) must therefore be incorporated in the new QSPP instance starting from node v. This is easily achieved by summing the row and the column of the quadratic cost matrix corresponding to arc (s', v) to the linear costs vector of the new problem.

5.2 Reduction Test

The size of the QSPP instance may be considerably reduced by eliminating the variables which do not appear in any optimal solution of a given instance. This can be done by considering the reduced costs associated with each arc in the lower bound computation at a given Branch-and-Bound node. For this, consider an incumbent solution x of value z, and Let ℓ_{ij} be a lower bound on the QSPP obtained when imposing the additional constraint $x_{ij} = 1$. If $\ell_{ij} \geq z$, then we can fix x_{ij} to zero. Note that the value of ℓ_{ij} can be computed as the sum of the current bound and the reduced cost associated to an arc (i, j).

This reduction test, in principle, can be performed for all arcs at each Branch-and-Bound node. However, in our computational experiments we found it more convenient to apply it at the root node and then at some small subset of nodes in the Branch-and-Bound tree.

6 Computational Results

In this section we present our computational experiments with the MILP formulation and the Branch-and-Bound algorithm introduced in this paper. We compare our methods with Cplex 12.6 when applied directly to the problem formulation (1). We also use Cplex 12.6 with default parameter settings to solve the MILP formulation. We implemented the algorithms in C++ and ran them on an Intel Xeon CPU E5335 (2 quad core CPUs with 2GHz). In the following, we first present the test instances and then provide the results in detail.

6.1 Test Instances

To evaluate and compare the approaches studied in this paper, we consider two groups of instances described as follows:

Grid1 Consists of a single class of grid-like networks with $k \times k$ nodes and $2k(k-1)$ arcs, for $k = 10, \ldots, 15$. Each node is linked by an arc to the node to the right and to the node above. The source node s is the node in the lower left corner of the grid, and the target node t is in the upper right corner. The linear and quadratic costs are generated uniformly at random in $\{1, \ldots, 10\}$.

Grid2 Consists of three classes of grid-like networks with a stricter scheme [13]. Each network consists of transshipment nodes forming a grid of n_r rows and n_c columns as well as a source node s and a target node t. The source node s is connected to the nodes of the first column, and the nodes of the last column are connected to the target node t. Each transshipment node is connected to the node on the right and to the node below if these exist. We randomly generate the linear and quadratic costs uniformly in $\{1, \ldots, 10\}$. Based on different values for n_r and n_c, we consider three classes: Grid2Square with $n_r = n_c \in \{16, 23\}$, Grid2Long with $n_r = 16,\ n_c \in \{32, 64\}$, and Grid2Wide with $n_r \in \{32, 64\},\ n_c = 16$.

6.2 Results

Tables 2 to 5 present the results. In each table, the first three columns give the number of nodes (n), number of arcs (m), and the optimal objective values (opt.) obtained by our Branch-and-Bound algorithm. The next columns present the results of CPLEX applied to the problem formulation (1) (Cplex(QP)), CPLEX applied to the MILP formulation (Cplex(MILP)), and the Branch-and-Bound algorithm. For each algorithm, we present the lower bound in the root node (lb_{root}), the total number of nodes enumerated in the search tree (nodes), and the total required time (in seconds) to solve the problem (time). An entry "TL" indicates that the corresponding algorithm was not able to solve the instance within the specified time limit. We considered a time limit of 10800 seconds for each instance. The lower bound in the root node of our Branch-and-Bound algorithm is the reformulation based bound. Note that all graphs in our test bed are acyclic, hence, we used the minimum flow formulation to find the parameters w_{ij} needed for the MILP formulation.

Table 2 reports the results for Grid1 instances. As we can observe, the bounds obtained by the reformulation scheme are stronger than those obtained by Cplex(QP) and Cplex(MILP). Moreover, the bounds obtained by Cplex(MILP) are much stronger than those of Cplex(QP). More precisely, the bounds produced by the reformulation scheme are, on average, 16.2% stronger than those produced by Cplex(MILP) and 64.6% stronger than those obtained by Cplex(QP). In addition, the bounds obtained by Cplex(MILP) are, on average, 57% stronger than Cplex(QP). Concerning the overall performance for solving the instances to optimality, both Cplex(MILP) and the Branch-and-Bound algorithm could solve all instances within the time limit while Cplex(QP) reached its limit for $n = 225$. When all approaches are able to solve an instance to optimality within the time limit, the Branch-and-Bound algorithm is, on average, about 160 times faster than Cplex(MILP) and 370 times faster than Cplex(QP). Also comparing Cplex(QP) and Cplex(MILP), the latter is about 2.3 faster than the former.

Tables 3 to 5 report the results for the Grid2Square, Grid2Long, and Grid2Wide instances, respectively. Again, the bounds obtained by the reformulation scheme are stronger than those obtained by Cplex(QP) and Cplex(MILP). For these instances, Cplex(QP) either reached its limit even in the root node or produced negative bounds. Such negative bounds can arise because Cplex(QP) needs to convexify the instances first, in order to obtain a tractable continuous relaxation. The convexification often leads to rather weak lower bounds, which in our case may even become negative. In fact, Cplex(QP) was able to solve to optimality only Grid2Square instances with $n = 258$ within the time limit. Concerning the overall performance of Cplex(MILP) and the Branch-and-Bound algorithm, the latter always outperforms the former and could solve all instances to optimality. In particular, for the Grid2Long instances, which seems to be the most difficult test set among the Grid2 instances, Cplex(MILP) was not able to solve to optimality instances with $n = 1026$ within the time limit while the Branch-and-Bound algorithm was able to solve all instances in less than 15 minutes.

Tables 3 to 5 report the results for the Grid2Square, Grid2Long, and Grid2Wide instances, respectively. Again, the bounds obtained by the reformulation scheme are stronger than those obtained by Cplex(QP) and Cplex(MILP). For these instances, Cplex(QP) either reached its limit even in the root node or produced negative bounds. Such negative bounds can arise because Cplex(QP) needs to convexify the instances first, in order to obtain a tractable continuous relaxation. The convexification often leads to rather weak lower bounds, which in our case may even become negative. In fact, Cplex(QP) was able to solve to optimality only Grid2Square instances with $n = 258$ within the time limit. Concerning the overall performance of Cplex(MILP) and the Branch-and-Bound algorithm, the latter always outperforms the former and could solve all instances to optimality. In particular, for the Grid2Long instances, which seems to be the most difficult test set among the Grid2 instances, Cplex(MILP) was not able to solve to optimality instances with $n = 1026$ within the time limit while the Branch-and-Bound algorithm was able to solve all instances in less than 15 minutes. For the instances of Grid2Square and Grid2Long for which both Cplex(MILP) and the Branch-and-Bound algorithm solve the problem to optimality within the time limit, the Branch-and-Bound algorithm is, on average, about 15 and 3.3 times faster than Cplex(MILP), respectively.

7 Conclusion

In this paper, we have studied the QSPP. We have shown that both the general QSPP and the AQSPP cannot be approximated unless $\mathsf{P} = \mathsf{NP}$. For the case of a convex objective function, we have presented an n-approximation algorithm, where n is the number of nodes in the graph, and showed that the problems is APX-hard. In order to solve the problem efficiently, we reformulated the problem as an MILP and solve it using a state-of-the-art solver. Moreover, we have proposed an exact Branch-and-Bound algorithm, where lower bounds are computed using a reformulation scheme. Our computational experiments indicate the power of our Branch-and-Bound algorithm. For the tested graph classes, it solves the QSPP to optimality considerably faster than the generic solver.

8 Acknowledgments

The first author has been supported by the German Research Foundation (DFG) under grant BU 2313/2. The second author is sponsored by the Air Force Office of Scientific Research, Air Force Material Command, USAF, under grant number FA8655-13-1-3066. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purpose not-withstanding any copyright notation thereon.

References

1. Amaldi, E., Galbiati, G., Maffioli, F.: On minimum reload cost paths, tours, and flows. Networks 57(3), 254–260 (2011)

2. Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., Protasi, M.: Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties. Springer (1999)
3. Berman, P., Fujito, T.: On approximation properties of the independent set problem for degree 3 graphs. In: Algorithms and Data Structures, Lecture Notes in Computer Science, vol. 955, pp. 449–460. Springer (1995)
4. Buchheim, C., Traversi, E.: Quadratic 0–1 optimization using separable underestimators. Tech. rep., Optimization Online (2015)
5. Caprara, A.: Constrained 0-1 quadratic programming: Basic approaches and extensions. European Journal of Operational Research 187(3), 1494–1503 (2008)
6. Carraresi, P., Malucelli, F.: A new lower bound for the quadratic assignment problem. Operations Research 40(1-supplement-1), 22–27 (1992)
7. Chen, B.Y., Lam, W.H.K., Sumalee, A., Li, Q., Shao, H., Fang, Z.: Finding reliable shortest paths in road networks under uncertainty. Networks and Spatial Economics 13(2), 123–148 (2012)
8. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische Mathematik 1(1), 269–271 (1959)
9. Gabow, H., Maheshwari, S.N., Osterweil, L.: On two problems in the generation of program test paths. Software Engineering, IEEE Transactions on SE-2(3), 227–231 (Sept 1976)
10. Gerschgorin, S.: Über die Abgrenzung der Eigenwerte einer Matrix. Izvestija Akademii Nauk SSSR, Serija Matematika 7(3), 749–754 (1931)
11. Gilmore, P.C.: Optimal and suboptimal algorithms for the quadratic assignment problem. Journal of the Society for Industrial & Applied Mathematics 10(2), 305–313 (1962)
12. Glover, F.: Improved linear integer programming formulations of nonlinear integer problems. Management Science 22(4), 455–460 (1975)
13. Kovács, P.: Minimum-cost flow algorithms: An experimental evaluation. Optimization Methods and Software 30(1), 94–127 (2015)
14. Lawler, E.L.: The quadratic assignment problem. Management science 9(4), 586–599 (1963)
15. Murakami, K., Kim, H.S.: Comparative study on restoration schemes of survivable atm networks. In: INFOCOM'97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE. vol. 1, pp. 345–352. IEEE (1997)
16. Öncan, T., Punnen, A.P.: The quadratic minimum spanning tree problem: A lower bounding procedure and an efficient search algorithm. Computers & Operations Research 37(10), 1762–1773 (2010)
17. Rostami, B., Malucelli, F.: Lower bounds for the quadratic minimum spanning tree problem based on reduced cost computation. Computers & Operations Research 64, 178–188 (2015)
18. Rostami, B., Malucelli, F., Frey, D., Buchheim, C.: On the quadratic shortest path problem. In: Bampis, E. (ed.) Experimental Algorithms, Lecture Notes in Computer Science, vol. 9125, pp. 379–390. Springer International Publishing (2015)
19. Sen, S., Pillai, R., Joshi, S., Rathi, A.K.: A mean-variance model for route guidance in advanced traveler information systems. Transportation Science 35(1), 37–49 (2001)
20. Sivakumar, R.A., Batta, R.: The variance-constrained shortest path problem. Transportation Science 28(4), 309–316 (1994)

Table 2: Results for the GRID1 instances. All times are given in seconds.

Instance			Cplex (QP)			Cplex (MILP)			B-and-B		
n	m	opt.	lb_{root}	nodes	time	lb_{root}	nodes	time	lb_{root}	nodes	time
100	180	621.0	200.0	7264	16.9	442.8	1943	2.8	511.0	601	0.3
100	180	635.0	211.0	8482	17.5	438.7	2687	3.7	512.0	1345	0.4
100	180	636.0	217.0	7078	15.6	452.5	2229	3.2	530.0	1159	0.3
100	180	661.0	209.0	11814	20.2	457.2	7332	15.8	534.0	1097	0.3
100	180	665.0	233.0	10974	20.6	468.3	7141	16.1	545.0	1145	0.4
121	220	813.0	253.0	33736	72.9	547.4	12135	35.5	663.0	1595	0.6
121	220	788.0	251.0	24883	61.9	539.1	9049	29.1	631.0	1767	0.7
121	220	795.0	225.0	26607	59.0	543.0	11211	33.0	645.0	1555	0.6
121	220	782.0	236.0	24863	62.4	544.0	9797	30.6	648.0	1335	0.6
121	220	767.0	228.0	19309	51.8	540.1	6111	20.2	644.0	2231	0.7
144	264	959.0	271.0	67971	203.8	640.6	26869	117.8	775.0	4555	1.8
144	264	963.0	282.0	91341	254.3	641.6	33383	157.1	764.0	7259	2.3
144	264	900.0	259.0	61308	209.1	615.6	15423	66.8	735.0	4991	1.9
144	264	960.0	236.0	104978	285.8	642.1	33939	152.2	766.0	5579	1.9
144	264	976.0	289.0	86862	249.8	654.5	33710	141.4	772.0	5651	2.0
169	312	1159.0	335.0	338092	1367.2	747.7	140710	727.6	891.0	14739	5.5
169	312	1178.0	333.0	342119	1315.2	765.4	119759	636.0	920.0	10145	4.3
169	312	1164.0	325.0	305351	1218.8	751.9	133369	750.6	876.0	13957	5.6
169	312	1110.0	301.0	231176	951.6	746.7	79201	458.7	875.0	6745	3.0
169	312	1115.0	322.0	175669	816.5	757.6	37872	211.8	897.0	8865	3.7
196	364	1363.0	364.0	1021928	5857.6	863.8	362553	2699.4	1064.0	25585	11.4
196	364	1367.0	357.0	1104406	6276.7	876.5	361179	2541.6	1056.0	27881	11.9
196	364	1320.0	334.0	715390	4171.6	841.2	216562	1586.1	1009.0	18447	8.3
196	364	1347.0	348.0	918668	5087.0	876.3	284703	2017.9	1062.0	16923	9.3
196	364	1344.0	354.0	835595	4706.4	878.9	278683	2105.2	1043.0	28473	12.2
225	420	1551.0	367.0	1539600	TL	989.4	405943	3598.9	1200.0	20395	10.7
225	420	1588.0	412.0	1707723	TL	1003.4	464441	3600.6	1211.0	55001	55.5
225	420	1561.0	419.0	1787478	TL	953.6	485195	3710.1	1168.0	88461	95.6
225	420	1569.0	386.0	1769978	TL	966.4	485644	3650.3	1146.0	47169	57.5
225	420	1582.0	389.0	1699500	TL	1001.7	471452	3689.5	1203.0	36603	35.0

Table 3: Results for the Grid2Square instances. All times are given in seconds.

Instance			Cplex (QP)			Cplex (MILP)			B-and-B		
n	m	opt.	lb_{root}	nodes	time	lb_{root}	nodes	time	lb_{root}	nodes	time
258	512	622	-330.0	9747	1951.3	530.6	161	4.3	593.6	89	3.9
258	512	632	-333.1	10599	2357.2	530.8	235	5.6	588.9	123	4.2
258	512	650	-334.7	14249	2866.3	530.6	309	6.4	564.6	99	3.8
258	512	641	-333.9	13720	1525.7	514.5	295	5.7	586.0	91	4.3
258	512	593	-329.6	8533	1749.5	521.9	74	3.7	562.8	49	3.6
531	1058	1283	-759.4	4684	TL	997.9	5579	518.6	1125.6	414	22.1
531	1058	1281	-757.0	4783	TL	1001.3	4899	492.9	1146.4	438	22.2
531	1058	1302	-812.7	4688	TL	1007.4	4944	490.1	1130.3	768	25.5
531	1058	1283	-757.8	5419	TL	979.2	5113	526.3	1129.0	568	27.1
531	1058	1263	-807.4	3354	TL	1009.2	2101	125.9	1132.3	314	27.8

Table 4: Results for the Grid2Long instances. All times are given in seconds.

Instance			Cplex (QP)			Cplex (MILP)			B-and-B		
n	m	opt.	lb_{root}	nodes	time	lb_{root}	nodes	time	lb_{root}	nodes	time
514	1008	2469	-254.1	57932	TL	1880.7	16655	1575.6	2140.5	1232	24.5
514	1008	2518	-318.7	72043	TL	1901.6	18219	1523.0	2145.2	1525	25.0
514	1008	2453	-254.8	64599	TL	1880.6	14214	1226.8	2134.5	1283	23.0
514	1008	2400	-250.9	95704	TL	1866.3	9880	861.7	2120.7	532	21.7
514	1008	2453	-243.3	100000	TL	1889.6	9104	825.3	2184.2	1025	21.8
1026	2000	9392	TL	TL	TL	7300.4	16564	TL	8308.8	11943	263.9
1026	2000	9521	TL	TL	TL	7285.4	14398	TL	8281.7	28518	525.9
1026	2000	9514	TL	TL	TL	7345.9	19723	TL	8264.7	30169	546.3
1026	2000	9546	TL	TL	TL	7299.5	16174	TL	8335.8	45043	750.3
1026	2000	9542	TL	TL	TL	7318.2	11747	TL	8381.2	34461	631.2

Table 5: Results for the Grid2Wide instances. All times are given in seconds.

Instance			Cplex (QP)			Cplex (MILP)			B-and-B		
n	m	opt.	lb_{root}	nodes	time	lb_{root}	nodes	time	lb_{root}	nodes	time
514	1040	633	TL	TL	TL	514.2	500	41.9	572.1	259	20.9
514	1040	621	TL	TL	TL	501.3	631	46.4	567.2	187	26.6
514	1040	605	TL	TL	TL	512.2	383	38.0	585.1	63	27.0
514	1040	645	TL	TL	TL	512.6	921	61.0	569.0	479	22.5
514	1040	604	TL	TL	TL	496.2	406	40.1	559.1	321	23.4
1026	2096	633	TL	TL	TL	499.0	1723	436.6	562.3	403	107.0
1026	2096	620	TL	TL	TL	507.6	1193	363.7	574.8	245	109.8
1026	2096	631	TL	TL	TL	504.8	1416	376.8	581.0	299	101.5
1026	2096	639	TL	TL	TL	497.6	2138	516.4	574.5	455	103.9
1026	2096	602	TL	TL	TL	496.7	776	261.7	567.2	137	119.1