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

Borzou Rostami[a,*], André Chassein[b], Michael Hopf[b], Davide Frey[d], Christoph Buchheim[c], Federico Malucelli[e], Marc Goerigk[f]

[a] *École de Technologie Supérieure de Montréal and Interuniversity Research Center on Enterprise Networks, Logistics and Transportation (CIRRELT), Canada*
[b] *Fachbereich Mathematik, TU Kaiserslautern, Germany*
[c] *Fakultät für Mathematik, TU Dortmund, Germany*
[d] *INRIA-Rennes Bretagne Atlantique, Rennes, France*
[e] *Dipartimento di Elettronica e Informazione, Politecnico di Milano, Italy*
[f] *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 P = 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 P = 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: Combinatorial optimization, 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

*Corresponding author. *E-mail:* bo.rostami@gmail.com
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 notwithstanding any copyright notation thereon.

cost of the path [23], and problems in which additional costs arise from pairs of arcs in a solution [1, 13].

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 $x^\top Q x + c^\top x$ 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 [23] 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. In [22], the authors 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. In [18], the authors 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 in [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

	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.

approximated 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 [21], 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

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

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, 21, 13]. 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 [21].

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 $\mathcal{L} = \{(a_1, \overline{a}_1), \ldots, (a_k, \overline{a}_k)\}$. 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 3.1. *The QSPP cannot be approximated unless P = 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 $\mathcal{L}$. If and only if the instance of the PFPP problem is a yes-instance there exists a solution of the QSPP with objective value 0. Contrary, if the instance of the PFPP is a no-instance each feasible solution of

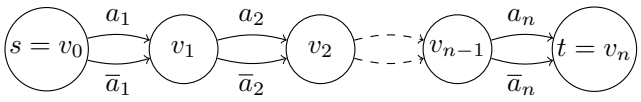

Figure 1: The graph used for the reduction in the proof of Theorem 3.2.

the QSPP has cost of at least 2. Hence, every approximation algorithm for the QSPP could be used to decide the PFPP, which implies the non approximability result. □

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 3.1.

Theorem 3.2. *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'.

In the proof, 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. Function $\mathfrak{g}$ has three inputs: An error parameter ϵ, an instance $\mathcal{I}$, and an $(1 + \mathfrak{h}(\epsilon))$-approximate solution of the corresponding problem $\mathfrak{f}(\mathcal{I})$. The output of $\mathfrak{g}$ is a solution of $\mathcal{I}$ that is at most $(1 - \epsilon)$ times worse than the optimal solution.

Proof. 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 defined as the following multiset

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

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.
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 them (with out loss of generality the node with the smaller index). 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

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

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

Note that the objective function of the QSPP is given by the expression $x^TQx + c^Tx$, which can be simplified to $x^T(Q + \mathrm{Diag}\,(c))x$, where $\mathrm{Diag}\,(c)$ 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.

Proof. As explained above, we assume that the objective function of the QSPP is given by $f(x) = x^TMx$ with a quadratic matrix M. 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 $x^T\mathrm{Diag}\,(d)x = d^Tx$ for all binary x.

We now prove the approximation guarantee of g. As all entries of the matrix M are positive, we have that $g(x) \leq f(x) \ \forall x \geq 0$. To see that $f(x) \leq k \cdot g(x)$ consider the following (without loss of generality we assume that the first k entries of x are one):

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

The last inequality follows since M is positive semidefinite. To see that $0 \leq M_{ii} x_i^2 - 2M_{ij} x_i x_j + M_{jj} x_j^2$ for all i, j, consider $\hat{x} := e_i \cdot x_i - e_j \cdot x_j$, where e_i is the i^{th} unit vector. As f is a convex function, M must be positive semi definite. Hence, $0 \leq \hat{x}^T M \hat{x} = M_{ii} x_i^2 - 2M_{ij} x_i x_j + M_{jj} x_j^2$.

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. Independently, the same result has been proved in [14] using a reduction from the 2-disjoint path problem. Moreover, one can use a similar (but simpler) reduction to show that the QSPP cannot be approximated unless $\mathsf{P} = \mathsf{NP}$, even if the underlying graphs is series-parallel. The idea of the reduction is to use a chain of two parallel arcs, representing the literal assignment, followed by a chain of seven parallel arcs, representing the feasible-clause assignment. The consistency between literal and clause assignment can be ensured by the quadratic cost function.

Theorem 3.4. *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 $x_1, \ldots, x_n$ and m clauses $C_1, \ldots, C_m$. 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 $V = \{s\} \cup \{v_1, \ldots, v_n\} \cup \{C_1, \ldots, C_m\} \cup \{t\} \cup V'$ 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 $V' = \{v_{ijk}, \overline{v}_{ijk} | i \in \{1, \ldots, n\}, j \in \{1, \ldots, m\}, k \in \{1, \ldots, 7\}\}$ consists of $14mn$ vertices that are used to establish an individual connection between each clause and each literal. We connect v_{i-1} and v_i for $i = 1, \ldots, n$ $(v_0 := s)$ with two distinctive paths $P_i, \overline{P}_i$ of length $7m + 1$

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

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 $Q_{j1}, \ldots, Q_{j7}$ 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.

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 $Q_{j1}, \ldots, Q_{j7}$ 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

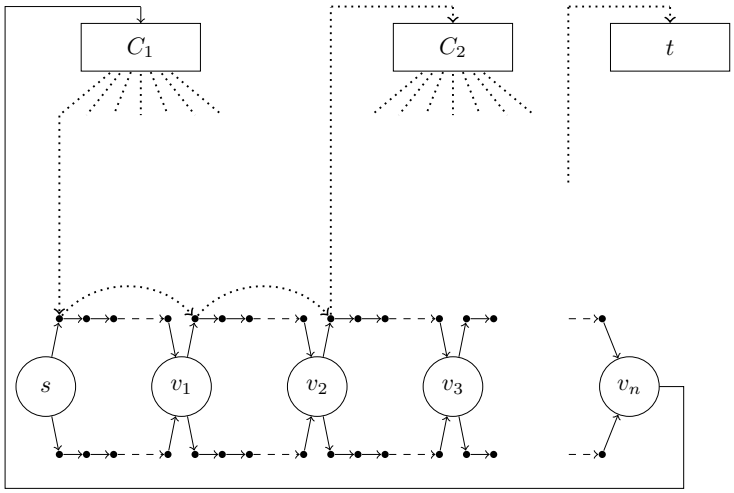

Figure 2: The graph used for the reduction in the proof of Theorem 3.4. All horizontal arcs pointing from left to right are of type one, all dotted arcs are of type two. All dashed arcs indicate chains of arcs. The dotted path that is completely shown corresponds to the assignment $x_1 = 0, x_2 = 0$, and $x_3 = 0$ for clause C_1.

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. Let x' be the literal assignment represented by P'. As the 3SAT instance is a no-instance, at least one clause can not be satisfied by x'. Let C_j be the clause which is not satisfied by x'. Since one of the seven paths $Q_{j1}, \ldots, Q_{j7}$ 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 $k = 1, \ldots, 7$ 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 3.4 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 in [11] and [16] to compute a lower bound for the Quadratic Assignment Problem (QAP). This approach is one of the best known for the QAP due to its simplicity and lower computational cost, and has been adapted to many other quadratic 0–1 problems in the meantime [5, 19].

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

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

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, 20]. To overcome this problem, we present an iterative procedure in the following subsection.

4.2. An Iterative Procedure to Improve the GL 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:

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

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

$$\tilde{c}_{ij} = c_{ij} + z_{ij} \tag{2}$$

$$\tilde{q}_{ijkl} = q_{ijkl} + (\lambda^*_{ij})_k - (\lambda^*_{ij})_l \quad \forall (k,l) \in A, (k,l) \neq (i,j) \tag{3}$$

$$\tilde{q}_{ijij} = q_{ijij} + (\lambda^*_{ij})_i - (\lambda^*_{ij})_j - \pi^*_{ij} \tag{4}$$

where z_{ij} is the optimal value of $\mathcal{P}_{ij}$ and 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.

$$\begin{array}{lll} \text{RQSPP:} & \overline{z}^* = \min & \displaystyle\sum_{(i,j),(k,l)\in A} \tilde{q}_{ijkl} x_{ij} x_{kl} + \sum_{(i,j)\in A} \tilde{c}_{ij} x_{ij} \\ & \text{s.t.} & x \in X_{st}, \; x \text{ binary.} \end{array} \tag{5}$$

Theorem 4.1. *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

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

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

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

□

The reformulation strategy incorporates the quadratic contributions captured by the GL bound into the linear costs. This makes it possible to compute the GL bound for the original problem by simply ignoring the quadratic costs in RQSPP and computing a linear shortest path. Applying the GL bound to the reformulated problem, on the other hand, provides no additional improvement as the GL procedure cannot increase the linear costs any further ($z_{ij}=0\ \forall(i,j)\in A$). However, it is possible to further improve the bound by directly changing the quadratic cost matrix. In Example 4.1 we present a small QSPP sample instance, which shows that two different cost matrices, specifying the same QSPP can lead to different GL bounds.

Example 4.1. *Consider an instance of the QSPP with the underlying graph depicted in Figure 3 and the cost structure shown in Table 2. We solve the subproblems for each edge and add the results to the linear costs of each edge. We present in Figure 4 the two different shortest path problems corresponding*

to cost matrices Q_1 and Q_2 which need to be solved to compute the GL bound. The left number over the edge represents its cost c_i and the right the optimal value z_i of the corresponding subproblem. The GL bound is found by solving a shortest path problem with the updated cost. Note that the cost matrices Q_1 and Q_2 lead to different subproblems with different optimal values. The shortest path on the left has length 5 and the shortest path on the right costs 6.

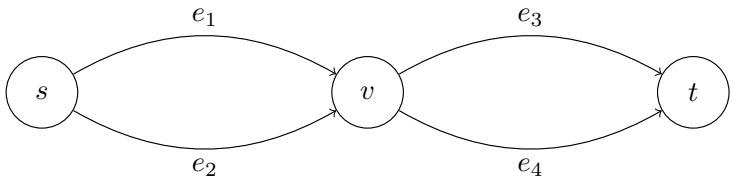

Figure 3: The underlying graph of the sample QSPP instance.

	c_i
e_1	1
e_2	2
e_3	2
e_4	1

Q_1	e_1	e_2	e_3	e_4
e_1	0	4	6	8
e_2	0	0	2	4
e_3	0	0	0	2
e_4	0	0	0	0

Q_2	e_1	e_2	e_3	e_4
e_1	0	2	3	4
e_2	2	0	1	2
e_3	3	1	0	1
e_4	4	2	1	0

Table 2: The cost structure of the instance shown in Figure 3. Note that both quadratic cost matrices Q_1 and Q_2 describe the same QSPP.

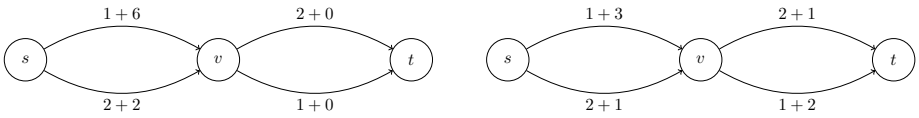

Figure 4: The figure on left corresponds to the cost matrix Q_1, while the right one corresponds to the cost matrix Q_2.

Based on the results of Theorem 4.1 and motivated by Example 4.1, we propose an iterative procedure to find a sequence of reformulations of the original problem that lead to better and better lower bounds. Note that an iterative procedure which sequential improves the lowerbound for the QAP was already introduced in [6]. Starting from the original reformulation RQSPP with linear costs $\tilde{c}$ and quadratic costs $\tilde{Q}$, the first iteration finds a new reformulation, RQSPP', with new linear cost $\tilde{c}'$ and quadratic cost $\tilde{Q}'$, which provides a better lower bound than the GL bound obtained with RQSPP. The iterative procedure is summarized as follows.

- *Step 1. Initialize $\tilde{c}_{ij} = c_{ij}$ for all $(i,j) \in A$, and $\tilde{q}_{ijkl} = q_{ijkl}$ for all $(i,j),(k,l) \in A$. Set an iteration counter $itr = 0$ and the current lower bound $LB^{itr} = 0$.*
- *Step 2. For each $(i,j) \in A$, change the coefficient $\tilde{q}_{ijkl}$ to a percentage of the sum of $\tilde{q}_{ijkl} + \tilde{q}_{klij}$, and then adjust $\tilde{q}_{klij}$ so that the sum remains constant. (In our computational experiments we set $\tilde{q}_{ijkl} = \tilde{q}_{klij} = (\tilde{q}_{ijkl} +$*

$\tilde{q}_{klij})/2$ *for all* $(i,j),(k,l) \in A$ *with* $i < k$*). Upon solving the corresponding subproblem* $(\mathcal{P}_{ij})$ *with new cost* $\tilde{Q}$*, use* (2) *to* (4) *to update* $\tilde{c}$ *and* $\tilde{Q}$ *and proceed to the next step.*

- *Step 3.* Solve a shortest path problem with new cost $\tilde{c}$, and set LB^{itr} equal to the objective value. Stop when a predetermined number of iterations have been performed, otherwise increase the iteration counter by 1 and return to *Step 2*.

4.3. An LP-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

$$z^* = \min \left\{ \sum_{(i,j)\in A} a_{ij}(x)x_{ij} : \ x \in X_{st} \right\}. \tag{6}$$

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

$$y_{ij} \geq (c_{ij} + z_{ij})x_{ij} \quad (i,j) \in A. \tag{7}$$

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:

$$y_{ij} \geq \sum_{(k,l)\in A} q_{ijkl}x_{kl} - w_{ij}(1 - x_{ij}) + c_{ij}x_{ij} \quad (i,j) \in A. \tag{8}$$

Using (7) and (8) 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 to 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 (an upper bound), and a method to partition the feasible region of a given problem (branching rule). Both the lower and upper bound can be generated by any of the lower bounding procedures we described in Sections 4.1 and 4.2, as their application also provides feasible QSPP solutions. The objective value of any feasible QSPP solution provides an upper bound to the QSPP.

To satisfy the third requirement, we instead exploit the structure of the QSPP. 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 therefore consists in considering the subproblems associated with each of the neighbors of the start node. The same idea can then be applied recursively by taking the considered neighbor as the start node of the subproblem, and considering its own neighbors.

Let us consider a sub-problem corresponding to a neighbor v. The solution to this sub-problem consists of the concatenation of the path from s to v and the solution to a quadratic shortest path problem from v to t. Clearly, to obtain a correct subproblem when forcing a neighbor, v, as a new start node, it is necessary to update the costs of the arcs in the subproblem by taking into account the presence of the arc forced into the solution by the branching operation.

Let us consider what happens when we partition a problem from u to t by branching on one of the neighbors of u, for example v. The branching step involves forcing arc (u, v) into the solution, and the solution to the problem from u to t will consist of the arc (u, v) together with the solution of a new subproblem from v to t. The new v-to-t subproblem will thus have to take into account the presence of arc (u, v) in the solution. To this end, the cost of each arc in the v-to-t subproblem will have to incorporate the quadratic contribution corresponding to its coexistence with arc (u, v). This is easily achieved by summing the row and the column of the quadratic cost matrix corresponding to arc (u, v) to the linear costs vector of the new problem. If $c^{u-\text{to}-t}$ and $q^{u-\text{to}-t}$ are the cost vector and quadratic cost matrix for the problem from u to t, and $c^{v-\text{to}-t}$ is the linear cost vector for the sub-problem from v to t, then we have the following, for each $(i, j) \in A$:

$$c_{ij}^{v-\text{to}-t} = c_{ij}^{u-\text{to}-t} + q_{ijuv}^{u-\text{to}-t} + q_{uvij}^{u-\text{to}-t}.$$

After updating the costs like this, the branch-and-bound algorithm first computes a lower and upper bound for the new subproblem, and then determines whether to close this branch (if the lower bound is greater than the current lowest upper bound), or to keep exploring updating the current best lowest upper bound if appropriate.

As described above, the lower and upper bounds can be obtained using any of the bounding algorithms described in Section 4. Yet the novelty of our solution consists in adopting a hybrid approach. Specifically, at the root node of the branching tree, we apply the iterative procedure described in Section

4.2 to obtain a RQSPP. Then at each node of the Branch-and-Bound tree, we simply obtain an upper and lower bound by solving a linear shortest path problem with the node's current linear cost vector. The linear cost of the path define the lower bound and the linear plus quadratic cost of the path define the upper bound. Before devising this hybrid approach, we had also run some experiments with Branch-and-Bound strategies that use the same bounding algorithm (e.g. the GL bound, or the reformulation) throughout all nodes, but the overall computing times were worse. While computing a new reformulation at all Branch-and-Bound nodes reduces the number of nodes, it increases the time required to compute the bound at each node, thereby leading to poorer overall performance.

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 Bound and Branch-and-Bound algorithms in C++ and ran them on an Intel Xeon CPU E5335 (2 quad core CPUs running at 2GHz). For the reformulation bound, we used a maximum of 20 iterations as a stopping condition. In the following, we first present the test instances, give some results showing the effectiveness of the iterative procedure described in Section 4.2, and provide the Branch-and-Bound results in detail.

6.1. Test Instances

To evaluate and compare the approaches studied in this paper, we consider three groups of instances, Grid1, Grid2, and Grid3, described as follows.

Grid1. This class consists of grid-like networks with $n = k \times k$ nodes and $m = 2k(k-1)$ 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. We consider three variants of Grid1 with the following cost structures.

- Grid1Dense: General QSPP with dense quadratic cost matrices Q. For linear costs, we associate each arc with a uniformly random integer in $\{1, \ldots, 10\}$. For quadratic costs, we associate each pair of arcs with a uniformly random integer in $\{0, \ldots, 9\}$.
- Grid1Sparse: General QSPP with sparse quadratic cost matrices Q. For linear costs, we associate each arc with a uniformly random integer in $\{1, \ldots, 10\}$. For quadratic costs, we associate each pair of adjacent arcs, as well as one third of non-adjacent pairs with a uniformly random integer in $\{0, \ldots, 9\}$. This represents the situation in which pair-wise interactions are more likely for adjacent arcs. Table 3 gives the average number of nonzero quadratic costs ($\#nzqc$) over five randomly generated instances for both sparse and dense matrices for $n \in \{100, 121, 144, 169, 196, 225\}$.

Table 3: Average numbers of the non-zero quadratic costs for the sparse and the dense instances

Instance		*dense*		*sparse*	
n	m	$\#nzqc$	$density(\%)$	$\#nzqc$	$density(\%)$
100	180	14483	89.4	5137	31.7
121	220	21675	89.6	7622	31.5
144	264	31221	89.6	10867	31.2
169	312	43648	89.6	15112	31.0
196	364	59452	89.7	20484	30.9
225	420	79157	89.7	27195	30.8

- GRID1CONVEX: The mean-variance Shortest Path problem with non-negative variances. We generate matrix Q as described in [17] for the Steiner travelling salesman problem with correlated costs: First we generate $|A| \times |A|$ non-negative random numbers from the standard normal distribution. Each of the $|A|$ vector coordinates are scaled to have length one and multiplied by a uniform random number in range $[0, 10]$. Let U be the matrix whose columns are the $|A|$ dimensional vectors, in any order, we set $Q = U^T U$. By construction matrix Q is positive semidefinite and the resulting QSPP instance is convex.

GRID2. To show how the topology of the graph can affect the level of difficulty of the instances, we generated a cyclic version of GRID1 with $n = k \times k$ nodes and $m = 4k(k-1)$ arcs, for $k = 10, \ldots, 15$. Each node is linked by two arcs to the nodes to the right and left as well as to the nodes above and below. To generate linear and quadratic costs, we proceed like for GRID1 for the arcs that were already in GRID1. For the new arcs that are only in GRID2 and for pairs of arcs that involve at least one new arc, we set the corresponding cost to 0.

GRID3. This class consists of three sub-classes of grid-like networks with a stricter scheme [15]. 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. Based on different values for n_r and n_c, we consider three classes: GRID3SQUARE with $n_r = n_c \in \{16, 23\}$, GRID3LONG with $n_r = 16,\ n_c \in \{32, 64\}$, and GRID3WIDE with $n_r \in \{32, 64\},\ n_c = 16$. For all instances, we generate linear and quadratic costs in the same way as for GRID1DENSE: uniformly at random in, respectively, $\{1, \ldots, 10\}$ and $\{0, \ldots, 9\}$.

6.2. Behavior of the Reformulation Bound

Before presenting the results of our solution method, we briefly analyze the behavior of our reformulation lower bound. Table 4 analyzes the improvement of the lower bound and the upper bound at each iteration on the largest instance of each instance class. The data shows that our iterative procedure significantly improves the GL bound in all instance classes. Moreover, it achieves this improvement in just a few iterations. In all considered instances, 9 iterations suffice

to obtain bound values that are very close to those obtained when our stopping condition of 20 iterations is met (last row). This suggests that, if needed, we could further improve computation time by stopping after fewer iterations or when the improvement in the value of the bound goes below a certain threshold.

Table 4: Bound values at each iteration for the largest instances of each instance class.

Iter.	Grid1Dense			Grid1Sparse			Grid1Convex			Grid2		
	lb	ub	time	lb	ub	time	lb	ub	time	lb	ub	time
0	104.0	1848	0.00	104.0	428	0.00	238.0	296	0.00	94.0	1825	0.00
1	977.5	1698	0.18	190.5	444	0.16	243.5	296	0.16	826.5	1792	0.55
2	1105.5	1738	0.33	204.7	444	0.31	247.0	296	0.31	926.7	1792	1.07
3	1151.8	1718	0.47	211.1	432	0.45	249.2	296	0.45	964.4	1792	1.60
4	1174.6	1738	0.61	215.4	432	0.60	250.9	296	0.59	982.2	1792	2.15
5	1185.5	1738	0.75	218.0	432	0.74	251.8	296	0.74	991.7	1792	2.71
6	1192.1	1718	0.89	219.3	432	0.89	252.3	296	0.88	997.1	1792	3.27
7	1195.9	1718	1.04	219.9	432	1.03	252.6	296	1.03	1000.2	1792	4.05
8	1198.3	1718	1.18	220.2	432	1.17	252.8	296	1.17	1002.0	1792	4.78
...	...	...	...	...	...	...	...	...	...	...	...	...
20	1202.9	1718	2.94	220.9	432	2.90	253.0	296	2.91	1005.3	1792	11.60

Iter.	Grid3Square			Grid3Long			Grid3Wide		
	lb	ub	time	lb	ub	time	lb	ub	time
0	95.0	1828	0.00	281.0	14020	0.02	53.0	853	0.02
1	966.0	1284	1.04	7244.5	9847	3.69	492.5	661	4.64
2	1060.2	1284	2.01	7940.5	9542	7.22	533.7	661	8.97
3	1096.1	1284	2.99	8174.4	9557	10.86	547.8	661	13.32
4	1113.8	1284	3.97	8281.2	9557	14.58	557.8	661	17.72
5	1122.2	1284	4.97	8331.1	9557	18.37	562.8	661	22.15
6	1126	1284	6.01	8354.2	9557	22.23	564.7	661	26.56
7	1128.3	1284	7.06	8366.5	9557	26.11	565.8	661	30.99
8	1129.7	1284	8.10	8373.3	9557	30.01	566.5	661	35.43
...	...	...	...	...	...	...	...	...	...
20	1132.3	1284	20.42	8381.2	9557	77.65	567.2	661	88.65

6.3. Branch-and-Bound Results

Tables 5 to 11 present the results for our solution approach. In each table, the first three columns give, for each instance, the number of nodes (n), the number of arcs (m), and the optimal objective value (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 our 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 bound based on the iterative procedure.

Table 5: Results for the GRID1 instances with dense quadratic cost matrices. 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	510.6	441	0.6
100	180	635.0	211.0	8482	17.5	438.7	2687	3.7	511.4	1154	0.7
100	180	636.0	217.0	7078	15.6	452.5	2229	3.2	529.8	1085	0.7
100	180	661.0	209.0	11814	20.2	457.2	7332	15.8	533.6	1083	0.7
100	180	665.0	233.0	10974	20.6	468.3	7141	16.1	544.6	963	0.7
121	220	813.0	253.0	33736	72.9	547.4	12135	35.5	662.7	1637	1.1
121	220	788.0	251.0	24883	61.9	539.1	9049	29.1	630.7	1477	1.1
121	220	795.0	225.0	26607	59.0	543.0	11211	33.0	644.8	1336	1.0
121	220	782.0	236.0	24863	62.4	544.0	9797	30.6	647.2	813	1.0
121	220	767.0	228.0	19309	51.8	540.1	6111	20.2	643.4	2068	1.7
144	264	959.0	271.0	67971	203.8	640.6	26869	117.8	774.7	3457	2.2
144	264	963.0	282.0	91341	254.3	641.6	33383	157.1	763.0	6257	3.2
144	264	900.0	259.0	61308	209.1	615.6	15423	66.8	734.2	3198	2.1
144	264	960.0	236.0	104978	285.8	642.1	33939	152.2	765.5	4321	2.6
144	264	976.0	289.0	86862	249.8	654.5	33710	141.4	771.3	4389	2.6
169	312	1159.0	335.0	338092	1367.2	747.7	140710	727.6	890.2	11036	6.2
169	312	1178.0	333.0	342119	1315.2	765.4	119759	636.0	919.5	8223	4.7
169	312	1164.0	325.0	305351	1218.8	751.9	133369	750.6	875.1	9374	5.0
169	312	1110.0	301.0	231176	951.6	746.7	79201	458.7	874.1	5124	3.8
169	312	1115.0	322.0	175669	816.5	757.6	37872	211.8	896.1	7175	4.5
196	364	1363.0	364.0	1021928	5857.6	863.8	362553	2699.4	1045.6	18397	12.0
196	364	1367.0	357.0	1104406	6276.7	876.5	361179	2541.6	1055.5	22936	14.1
196	364	1320.0	334.0	715390	4171.6	841.2	216562	1586.1	1008.4	14530	8.9
196	364	1347.0	348.0	918668	5087.0	876.3	284703	2017.9	1061.4	13739	9.3
196	364	1344.0	354.0	835595	4706.4	878.9	278683	2105.2	1042.8	22825	13.8
225	420	1551.0	367.0	1539600	TL	989.4	405943	3598.9	1199.7	15070	13.2
225	420	1588.0	412.0	1707723	TL	1003.4	464441	3600.6	1210.2	40451	29.5
225	420	1561.0	419.0	1787478	TL	953.6	485195	3710.1	1167.8	62190	42.2
225	420	1569.0	386.0	1769978	TL	966.4	485644	3650.3	1145.7	39097	30.3
225	420	1582.0	389.0	1699500	TL	1001.7	471452	3689.5	1202.9	27193	18.4

Tables 5, 6, and 7 report the results for the three variants of the GRID1 instances. Table 5 shows that our reformulation scheme achieves stronger root bound than Cplex(QP) and Cplex(MILP). In turn, Cplex(MILP) obtains much stronger bounds than Cplex(QP). More precisely, the bounds produced by our 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 those of Cplex(QP).

Concerning the overall performance for solving the instances to optimality, both Cplex(MILP) and our Branch-and-Bound algorithm can solve all instances within the time limit while Cplex(QP) reaches the three-hour limit for $n = 225$. When all approaches are able to solve an instance to optimality within the time limit, our reformulation-based Branch-and-Bound algorithm does so about 140 times faster than Cplex(MILP) and 320 times faster than Cplex(QP). Also Cplex(MILP) reaches optimality about 2.3 times faster than Cplex(QP).

Table 6: Results for the GRID1 instances with sparse quadratic cost matrices. 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	179.0	142.5	252	8.6	119.8	1244	0.9	124.8	955	0.7
100	180	184.0	141.9	186	8.9	126.3	879	0.8	136.2	573	0.6
100	180	193.0	160.6	166	6.6	139.7	917	0.8	148.8	529	0.6
100	180	192.0	157.4	215	8.5	132.8	1769	1.1	142.3	866	0.6
100	180	193.0	160.3	166	11.4	139.0	1356	1.2	145.4	1213	0.7
121	220	223.0	169.3	515	21.1	147.3	6728	8.2	155.9	891	1.0
121	220	207.0	163.4	225	16.1	140.3	2155	2.1	131.6	1168	1.1
121	220	222.0	177.1	212	20.7	148.3	5664	6.5	167.9	2657	1.4
121	220	215.0	174.7	194	17.4	149.8	2365	2.1	170.0	613	0.9
121	220	218.0	173.7	338	18.5	148.9	2256	1.7	161.3	1280	1.0
144	264	246.0	183.6	795	36.5	161.3	6954	12.6	169.3	2604	1.9
144	264	234.0	178.8	484	31.3	144.6	7577	13.4	170.6	1907	1.8
144	264	225.0	169.9	627	36.0	146.9	7896	12.4	149.0	1839	1.8
144	264	238.0	176.0	679	39.0	145.6	7683	14.4	156.0	3975	2.3
144	264	248.0	189.8	581	35.6	162.8	8409	14.3	167.6	2055	1.7
169	312	276.0	207.4	836	74.9	180.4	12652	26.6	185.3	8569	5.0
169	312	283.0	204.3	1162	72.9	176.8	20471	36.8	179.0	2907	2.8
169	312	265.0	195.7	550	47.5	164.5	9452	20.8	170.3	1429	2.1
169	312	277.0	199.1	1794	87.8	168.9	22966	40.9	180.1	4563	3.5
169	312	273.0	199.8	1041	76.6	169.2	15733	31.6	179.5	5788	3.9
196	364	319.0	223.7	2239	203.4	179.1	56025	131.9	205.3	11965	7.6
196	364	334.0	232.4	6420	609.2	190.6	75795	169.2	226.3	13659	9.3
196	364	319.0	224.3	1574	164.6	184.8	52323	107.6	214.6	4640	4.6
196	364	311.0	218.6	1507	134.5	190.6	27301	68.9	206.9	6815	5.4
196	364	312.0	215.9	6748	592.9	177.8	84084	185.4	199.7	8889	6.4
225	420	368.0	244.7	7511	1098.0	212.9	154715	429.0	235.6	13828	11.9
225	420	370.0	238.4	10303	1472.7	201.3	253601	655.0	234.4	19433	16.2
225	420	336.0	222.3	7349	979.4	184.8	121651	368.4	203.4	30055	20.6
225	420	345.0	229.1	2704	277.5	192.5	123893	338.9	212.5	14317	11.1
225	420	361.0	241.8	6719	1054.8	206.1	217408	556.8	220.9	15493	12.7

We can observe from Table 6 that the sparse instances are much easier to solve: all the three approaches can solve all instances within the time limit. One interesting observation is that Cplex(QP) obtains stronger lower bounds than both Cplex(MILP) and our reformulation technique. Yet Cplex(QP) remains much slower in finding the optimal solutions.

With convex objective functions (Table 7), the bounds obtained by Cplex(QP) and Cplex(MILP) are comparable to those obtained by our reformulation scheme. But in terms of the overall running time, our reformulation-based Branch-and-Bound algorithm still outperforms the others.

Table 8 presents the results for the GRID2 instances. As we expected, the cyclic structure of the network makes the problem much harder: this class of instances turns out to be the most difficult among all the tested datesets. Cplex(QP) and Cplex(MILP) can only solve instances of up to 121 and 144 nodes respectively within the time limit while our Branch-and-Bound algorithm can solve all but two instances within the allotted 3 hours. For these two instances,

Table 7: Results for the GRID1 instances with Convex objective function. 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	2024.0	1887.5	54	3.3	1928.1	86	0.3	1957.0	175	0.6
100	180	2047.0	1905.0	64	3.3	1962.4	92	0.3	1971.5	115	0.4
100	180	2021.0	1939.6	5	3.3	1985.8	13	0.2	1987.2	37	0.3
100	180	1971.0	1842.1	58	3.3	1882.5	132	0.3	1887.5	137	0.4
100	180	2158.0	2002.4	77	3.3	2060.2	126	0.3	2080.2	141	0.4
121	220	1802.0	1656.5	103	7.6	1695.0	296	0.7	1712.8	141	0.6
121	220	1895.0	1723.8	166	7.6	1774.1	411	0.8	1807.3	171	0.6
121	220	1821.0	1691.7	109	7.4	1722.3	185	0.5	1732.3	169	0.6
121	220	1798.0	1606.3	357	8.0	1669.1	962	1.3	1719.2	187	0.6
121	220	1843.0	1686.5	87	7.4	1744.9	277	0.6	1759.3	101	0.5
144	264	1320.0	1214.4	54	17.4	1239.6	115	0.6	1233.4	107	1.0
144	264	1444.0	1284.2	367	19.6	1333.6	696	1.7	1339.8	285	1.0
144	264	1451.0	1264.5	700	18.7	1305.5	1638	2.7	1325.4	753	1.0
144	264	1457.0	1303.1	328	18.7	1337.2	636	1.4	1351.5	347	1.3
144	264	1478.0	1329.3	273	16.4	1371.4	486	1.2	1372.0	286	1.0
169	312	991.0	836.1	1279	38.0	867.4	4832	18.1	896.1	2371	2.6
169	312	1010.0	868.6	883	36.6	893.1	2170	4.6	915.2	468	1.8
169	312	1018.0	885.6	723	33.8	899.1	2934	6.2	920.2	643	1.4
169	312	1014.0	869.9	892	34.6	883.6	2114	4.5	914.0	799	1.7
169	312	1024.0	882.0	1936	42.4	901.8	7201	28.2	922.6	1001	1.7
196	364	592.0	485.1	965	33.2	509.0	2271	5.3	540.5	376	2.2
196	364	604.0	486.0	3825	50.6	506.7	8578	36.8	521.1	1331	2.7
196	364	591.0	502.3	538	29.0	516.5	1090	3.2	504.1	611	2.5
196	364	568.0	468.5	1187	33.7	475.6	4596	21.8	486.0	2235	3.0
196	364	591.0	473.1	1177	38.9	491.2	2716	6.1	517.3	735	2.1
225	420	327.0	240.4	10332	46.0	274.1	7868	27.6	268.7	4421	5.8
225	420	321.0	254.2	2537	37.8	281.1	1568	4.3	275.7	849	3.3
225	420	324.0	243.8	4847	35.6	275.7	4607	17.8	270.1	2139	4.6
225	420	316.0	245.5	3035	33.0	278.2	2043	4.8	269.5	1355	3.7
225	420	296.0	232.9	833	19.5	265.5	524	1.6	253.0	373	4.2

we also ran our branch-and-bound approach without time limits. For the first of these two instances, the unlimited run terminated in 3 hours and 8 minutes and confirmed that the upper bound obtained at the end of 3 hours (1567) was already equal to the optimal solution. For the second, the run terminated in slightly more than 3 hours and 18 minutes, with an optimal solution of 1539 versus an upper bound of 1580 at the end of three hours.

Tables 9 to 11 report the results for the GRID3SQUARE, GRID3LONG, and GRID3WIDE instances, respectively. Again, our reformulation scheme obtains stronger root bounds than Cplex(QP) and Cplex(MILP). For these instances, Cplex(QP) either reaches its limit even in the root node or produces negative bounds. Such negative bounds can arise because Cplex(QP) first needs to convexify the instances 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 GRID3SQUARE instances with $n = 258$ within the time limit. Our reformulation-

Table 8: Results for GRID2 instances with uniformly random quadratic cost matrices. All times are given in seconds. Two of the instances show the upper bound corresponding to the current best feasible solution at the end of the time limit for our Branch and Bound method.

Instance			Cplex (QP)			Cplex (MILP)			B-and-B		
n	m	opt.	lb_{root}	nodes	time	lb_{root}	nodes	time	lb_{root}	nodes	time
00	360	660.0	241.6	20828	989.4	405.4	392722	307.5	472.7	24048	12.8
100	360	649.0	254.0	13512	628.4	400.6	290760	259.9	459.9	9572	7.1
100	360	615.0	234.0	13547	629.1	395.2	96583	83.4	451.3	4905	4.2
100	360	671.0	248.5	16726	718.3	413.0	321709	244.5	493.3	16886	10.7
100	360	642.0	223.3	12581	578.2	394.5	221362	180.1	459.5	15110	8.9
121	440	765.0	252.1	48235	4553.3	495.3	210120	236.2	581.1	13870	11.3
121	440	790.0	252.7	60393	4666.1	486.1	663537	683.6	560.0	26045	18.7
121	440	786.0	240.7	72035	6554.5	477.0	1342489	1332.2	560.9	28647	21.2
121	440	843.0	243.9	84789	6626.9	502.6	3921407	3444.5	590.4	85430	56.5
121	440	801.0	249.8	91603	TL	478.6	1781839	2056.2	543.6	71993	47.4
144	528	948.0	291.1	61908	TL	559.4	7905508	9671.2	664.3	112343	94.1
144	528	967.0	297.5	64769	TL	573.6	2761107	3815.4	676.1	266409	220.8
144	528	951.0	272.1	74994	TL	560.3	8070885	10799.0	668.9	100773	85.1
144	528	975.0	285.0	68677	TL	583.1	2061121	2765.2	687.0	81002	68.9
144	528	965.0	285.1	61244	TL	578.2	7824617	9658.1	693.2	142760	117.8
169	624	1154.0	309.8	38369	TL	669.8	8185752	TL	795.3	889716	929.5
169	624	1139.0	315.6	29091	TL	655.4	7794825	TL	778.7	442314	480.4
169	624	1138.0	283.1	18044	TL	659.0	7254006	TL	783.4	701955	756.7
169	624	1130.0	290.4	23762	TL	653.5	6042951	TL	754.3	369189	404.0
169	624	1143.0	307.4	18554	TL	666.5	6748938	TL	771.6	543761	603.7
196	728	1349.0	342.8	15411	TL	762.7	5582520	TL	925.4	840045	1249.6
196	728	1348.0	318.0	12201	TL	748.7	5201814	TL	887.0	1915627	2742.1
196	728	1326.0	299.2	16011	TL	730.4	5868876	TL	883.0	1317825	1886.7
196	728	1357.0	323.8	11751	TL	758.8	5949645	TL	910.9	1791916	2534.9
196	728	1294.0	331.2	17031	TL	762.4	5350713	TL	899.5	645206	922.9
225	840	1584.0	329.6	9141	TL	877.3	5175612	TL	1067.3	4621981	8953.0
225	840	1567.0*	322.0	11292	TL	862.9	4576566	TL	1045.4	5528272	TL
225	840	1512.0	340.6	6768	TL	831.7	4541850	TL	1003.5	2143568	4144.2
225	840	1580*	337.6	9486	TL	853.4	4385241	TL	1038.1	5534067	TL
225	840	1527.0	350.0	10323	TL	854.3	3992073	TL	1005.3	3179855	6120.3

* Upper bound given by the best feasible solution found by our Branch-and-Bound approach within the time limit.

based Branch-and-Bound algorithm instead solved all instances to optimality and outperformed even Cplex(MILP). In particular, for GRID3LONG instances, Cplex(MILP) was unable to solve instances with $n = 1026$ within the time limit while our Branch-and-Bound algorithm was able to solve all instances in less than 15 minutes. For the instances of GRID3SQUARE, GRID3WIDE, and GRID3LONG for which both Cplex(MILP) and the Branch-and-Bound algorithm solve the problem to optimality within the time limit, our Branch-and-Bound algorithm is, respectively, 15.9, 3.8, and 51.3 times faster than Cplex(MILP) on average. Finally, we observe that GRID3LONG instances appear to be the most difficult among the GRID3 groups, possibly because the optimal paths in

Table 9: Results for the Grid3Square 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	622.0	-330.0	9747	1951.3	530.6	161	4.3	594.0	89	4.3
258	512	632.0	-333.1	10599	2357.2	530.8	235	5.6	589.0	123	4.5
258	512	650.0	-334.7	14249	2866.3	530.6	309	6.4	564.7	99	4.5
258	512	641.0	-333.9	13720	1525.7	514.5	295	5.7	586.0	91	4.4
258	512	593.0	-329.6	8533	1749.5	521.9	74	3.7	562.8	49	4.3
531	1058	1283.0	-759.4	4684	TL	997.9	5579	518.6	1125.6	414	22.6
531	1058	1281.0	-757.0	4783	TL	1001.3	4899	492.9	1146.4	438	22.3
531	1058	1302.0	-812.7	4688	TL	1007.4	4944	490.1	1130.3	768	24.3
531	1058	1283.0	-757.8	5419	TL	979.2	5113	526.3	1129.0	568	23.0
531	1058	1263.0	-807.4	3354	TL	1009.2	2101	125.9	1132.3	314	22.9

Table 10: Results for the Grid3Long 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	2469.0	-254.1	57932	TL	1880.7	16655	1575.6	2140.5	1234	23.7
514	1008	2518.0	-318.7	72043	TL	1901.6	18219	1523.0	2145.2	1525	25.4
514	1008	2453.0	-254.8	64599	TL	1880.6	14214	1226.8	2134.5	1283	24.2
514	1008	2400.0	-250.9	95704	TL	1866.3	9880	861.7	2120.7	535	20.8
514	1008	2453.0	-243.3	100000	TL	1889.6	9104	825.3	2184.2	1025	23.0
1026	2000	9392.0	TL	TL	TL	7300.4	16564	TL	8308.9	11943	263.4
1026	2000	9434.0	TL	TL	TL	7285.4	14398	TL	8281.7	28520	525.8
1026	2000	9514.0	TL	TL	TL	7345.9	19723	TL	8264.7	30169	539.3
1026	2000	9520.0	TL	TL	TL	7299.5	16174	TL	8335.8	45043	747.4
1026	2000	9542.0	TL	TL	TL	7318.2	11747	TL	8381.2	34461	572.8

Grid3Long are on average twice as long as those in the other instance types.

7. Conclusion

In this paper, we studied the QSPP. We showed 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 presented an n-approximation algorithm, where n is the number of nodes in the graph, and we showed that the problem is APX-hard. In order to solve the problem efficiently, we provided two methods. First we reformulated the QSPP as an MILP and solved it using a state-of-the-art solver. Second, we proposed a lower bound based on an iterated reformulation approach. We used this lower bound to implement an exact Branch-and-Bound algorithm that can reach the optimal solution significantly faster than the state-of-the-art solver. One possible future research direction is to extend our approach for the QSPP with possible negative quadratic costs. In spite of the correctness of our reformulation strategy in this case, neither solving the linear part of the reformulated problem nor the GL procedure result in a valid lower bound for the original problem. Therefore, it would be interesting to extend our results with new bounding procedures that can lift this limitation.

Table 11: Results for the GRID3WIDE 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	633.0	TL	TL	TL	514.2	500	41.9	572.1	259	20.8
514	1040	621.0	TL	TL	TL	501.3	631	46.4	567.2	201	20.4
514	1040	605.0	TL	TL	TL	512.2	383	38.0	585.1	65	21.1
514	1040	645.0	TL	TL	TL	512.6	921	61.0	569.0	479	21.6
514	1040	604.0	TL	TL	TL	496.2	406	40.1	559.1	321	21.1
1026	2096	633.0	TL	TL	TL	499.0	1723	436.6	562.3	405	93.9
1026	2096	620.0	TL	TL	TL	507.6	1193	363.7	574.8	245	91.9
1026	2096	631.0	TL	TL	TL	504.8	1416	376.8	581.0	299	94.5
1026	2096	639.0	TL	TL	TL	497.6	2138	516.4	574.5	459	95.0
1026	2096	602.0	TL	TL	TL	496.7	776	261.7	567.2	145	89.5

[1] Edoardo Amaldi, Giulia Galbiati, and Francesco Maffioli. On minimum reload cost paths, tours, and flows. *Networks*, 57(3):254–260, 2011.

[2] Giorgio Ausiello, Pierluigi Crescenzi, Giorgio Gambosi, Viggo Kann, Alberto Marchetti-Spaccamela, and Marco Protasi. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. Springer, 1999.

[3] Piotr Berman and Toshihiro Fujito. On approximation properties of the independent set problem for degree 3 graphs. In *Algorithms and Data Structures*, volume 955 of *Lecture Notes in Computer Science*, pages 449–460. Springer, 1995.

[4] Christoph Buchheim and Emiliano Traversi. Quadratic 0–1 optimization using separable underestimators. Technical report, Optimization Online, 2015.

[5] Alberto Caprara. Constrained 0-1 quadratic programming: Basic approaches and extensions. *European Journal of Operational Research*, 187(3):1494–1503, 2008.

[6] Paolo Carraresi and Federico Malucelli. A new lower bound for the quadratic assignment problem. *Operations Research*, 40(1-supplement-1): S22–S27, 1992.

[7] Bi Y. Chen, William H. K. Lam, Agachai Sumalee, Qingquan Li, Hu Shao, and Zhixiang Fang. Finding reliable shortest paths in road networks under uncertainty. *Networks and Spatial Economics*, 13(2):123–148, 2012.

[8] Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[9] H. N. Gabow, Shachindra N. Maheshwari, and Leon J. Osterweil. On two problems in the generation of program test paths. *Software Engineering, IEEE Transactions on*, SE-2(3):227–231, Sept 1976.

[10] Semyon Gerschgorin. Über die Abgrenzung der Eigenwerte einer Matrix. *Izvestija Akademii Nauk SSSR, Serija Matematika*, 7(3):749–754, 1931.

[11] Paul C. Gilmore. Optimal and suboptimal algorithms for the quadratic assignment problem. *Journal of the Society for Industrial & Applied Mathematics*, 10(2):305–313, 1962.

[12] Fred Glover. Improved linear integer programming formulations of nonlinear integer problems. *Management Science*, 22(4):455–460, 1975.

[13] Laurent Gourvès, Adria Lyra, Carlos Martinhon, and Jérôme Monnot. The minimum reload s–t path, trail and walk problems. *Discrete Applied Mathematics*, 158(13):1404–1417, 2010.

[14] Hao Hu and Renata Sotirov. Special cases of the quadratic shortet path problem. Technical report, arXiv, 11 2016.

[15] Péter Kovács. Minimum-cost flow algorithms: An experimental evaluation. *Optimization Methods and Software*, 30(1):94–127, 2015.

[16] Eugene L. Lawler. The quadratic assignment problem. *Management science*, 9(4):586–599, 1963.

[17] Adam N. Letchford and Saeideh D. Nasiri. The Steiner travelling salesman problem with correlated costs. *European Journal of Operational Research*, 245(1):62 – 69, 2015.

[18] Kazutaka Murakami and Hyong S. Kim. 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*, volume 1, pages 345–352. IEEE, 1997.

[19] Temel Öncan and Abraham P. Punnen. The quadratic minimum spanning tree problem: A lower bounding procedure and an efficient search algorithm. *Computers & Operations Research*, 37(10):1762–1773, 2010.

[20] Borzou Rostami and Federico Malucelli. Lower bounds for the quadratic minimum spanning tree problem based on reduced cost computation. *Computers & Operations Research*, 64:178–188, 2015.

[21] Borzou Rostami, Federico Malucelli, Davide Frey, and Christoph Buchheim. On the quadratic shortest path problem. In Evripidis Bampis, editor, *Experimental Algorithms*, volume 9125 of *Lecture Notes in Computer Science*, pages 379–390. Springer International Publishing, 2015.

[22] Suvrajeet Sen, Rekha Pillai, Shirish Joshi, and Ajay K. Rathi. A mean-variance model for route guidance in advanced traveler information systems. *Transportation Science*, 35(1):37–49, 2001.

[23] Raj A. Sivakumar and Rajan Batta. The variance-constrained shortest path problem. *Transportation Science*, 28(4):309–316, 1994.