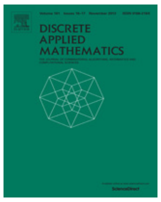
Discrete Applied Mathematics 166 (2014) 97–114

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam



Exact algorithms and heuristics for the Quadratic Traveling Salesman Problem with an application in bioinformatics[☆]

A. Fischer[a], F. Fischer[b], G. Jäger[c,*], J. Keilwagen[d,e], P. Molitor[f], I. Grosse[e,f,g]

[a] *Department of Mathematics, TU Dortmund, D-44227 Dortmund, Germany*

[b] *Department of Mathematics, Chemnitz University of Technology, D-09107 Chemnitz, Germany*

[c] *Department of Mathematics and Mathematical Statistics, University of Umeå, S-90187 Umeå, Sweden*

[d] *Institute for Biosafety in Plant Biotechnology, Julius-Kühn-Institut (JKI) - Federal Research Centre for Cultivated Plants, D-06484 Quedlinburg, Germany*

[e] *Department of Genebank, Leibniz Institute of Plant Genetics and Crop Plant Research (IPK), D-06466 Seeland OT Gatersleben, Germany*

[f] *Institute of Computer Science and Universitätszentrum Informatik, Martin Luther University Halle-Wittenberg, D-06120 Halle, Germany*

[g] *German Centre for Integrative Biodiversity Research (iDiv) Halle-Jena-Leipzig, D-04103 Leipzig, Germany*

A R T I C L E I N F O

Article history:
Received 21 September 2012
Received in revised form 1 July 2013
Accepted 24 September 2013
Available online 25 November 2013

Keywords:
Traveling Salesman Problem
Branch-and-bound
Branch-and-cut
Heuristical methods
Exact methods
Bioinformatics

A B S T R A C T

In this paper we introduce an extension of the Traveling Salesman Problem (TSP), which is motivated by an important application in bioinformatics. In contrast to the TSP the costs do not only depend on each pair of two nodes traversed in succession in a cycle but on each triple of nodes traversed in succession. This problem can be formulated as optimizing a quadratic objective function over the traveling salesman polytope, so we call the combinatorial optimization problem *quadratic TSP* (QTSP). Besides its application in bioinformatics, the QTSP is a generalization of the Angular-Metric TSP and the TSP with reload costs. Apart from the TSP with quadratic cost structure we also consider the related Cycle Cover Problem with quadratic objective function (QCCP). In this work we present three exact solution approaches and several heuristics for the QTSP. The first exact approach is based on a polynomial transformation to a TSP, which is then solved by standard software. The second one is a branch-and-bound algorithm that relies on combinatorial bounds. The best exact algorithm is a branch-and-cut approach based on an integer programming formulation with problem-specific cutting planes. All heuristical approaches are extensions of classic heuristics for the TSP. Finally, we compare all algorithms on real-world instances from bioinformatics and on randomly generated instances. In these tests, the branch-and-cut approach turned out to be superior for solving the real-world instances from bioinformatics. Instances with up to 100 nodes could be solved to optimality in about ten minutes.

1. Introduction

Given a weighted graph the *Traveling Salesman Problem* (TSP) is the problem of finding a tour with minimal costs where the costs are associated to each pair of nodes that are traversed in succession.The TSP is well-known to be an $\mathcal{NP}$-hard

[☆] A preliminary version of this paper appeared in the proceedings of the Second International Conference on Combinatorial Optimization and Applications (COCOA) 2008, Lecture Notes in Comput. Sci. 5165, pp. 211–224.

[*] Corresponding author. Tel.: +46 90 786 6141; fax: +46 90 786 5222.
E-mail address: gerold.jaeger@math.umu.se (G. Jäger).

problem. In this paper we consider an extension of the TSP. The *Quadratic Traveling Salesman Problem* (QTSP) is the problem of finding a cost minimal tour where the costs are associated to each *triple* of nodes that is traversed in succession. Because a path of three nodes is contained in a tour if and only if the two corresponding arcs are present, the QTSP can be modeled as optimizing a quadratic objective function over the traveling salesman polytope [8].

The QTSP is inspired by the problem of finding the optimal Permuted Markov (PM) model [10] or the optimal Permuted Variable Length Markov (PVLM) model [40] for a given set of DNA sequences. This problem is important in bioinformatics and genome research for the recognition of *transcription factor binding sites* and *splice sites*. Given a set of short DNA sequences of equal length n such as a set of *splice sites*, the task is to learn a pattern from this set that allows the recognition of further *splice sites* from a set of unknown sequences. Such patterns are typically learned by statistical models, and PM models and PVLM models are two of the most powerful models for the recognition of *splice sites*. Learning a pattern from data by a statistical model is often accomplished by finding the maximum likelihood estimates (MLEs) of the model parameters, and the problem of finding the MLEs of the parameters of PM models or PVLM models results in the QTSP. A more detailed description of the biological background and the connection to the QTSP is given in Section 2.

The TSP is closely related to the *Cycle Cover Problem* (CCP) that asks for a cost minimal set of disjoint cycles over all nodes. The CCP corresponds to a linear assignment problem, which can be solved by the Hungarian method in polynomial time [28]. However, we will show that its quadratic counterpart is $\mathcal{NP}$-hard. A special case of the QTSP is the *Angular-Metric TSP* [1], which for given n points in the Euclidean space aims to determine a tour with minimal total angle change. It has applications in robotics, *e.g.*, it allows to determine optimal robot paths w.r.t. energetic aspects. In particular, robots often tend to have a higher energy demand if the path bends sharply, so one is interested in tours without high curvature. An extension of Angle-TSP is the so called *TSP for Dubins vehicle* [38,33,32], where the task is to determine a shortest trajectory of a nonholonomic vehicle w.r.t. given curvature constraints. In their solution approach, the authors of [33] determine an optimal tour where the changes in direction are weighted against the length of the tour, which is a special case of QTSP, too. The TSP with reload costs is a further special case of QTSP. Here, given an arc-colored graph, the task is to determine a tour with minimal weighted sum of the color changes along the tour. Such cost structures arise, *e.g.*, in transport networks if the costs for loading processes are high in comparison to the transportation costs [2].

In this work, we present several exact and heuristical algorithms for the solution of the QTSP. We consider three exact algorithms. The first algorithm transforms the QTSP to an STSP to be solved by the state-of-the-art solver Concorde [7]. The second algorithm is a Branch-and-Bound algorithm based on combinatorial lower bounds. Here, we use a lower bound on the optimal value of the $\mathcal{NP}$-hard QCCP as a lower bound for QTSP. The third algorithm employs a Branch-and-Cut (BnC) algorithm based on a linear relaxation of a linearized integer programming formulation of the QTSP, which allows us to handle the well-known subtour elimination constraints [8]. Apart from these constraints we present further valid inequalities of the integer program that can be added during the BnC algorithm. Using the new cutting planes we could reduce the running times of the instances from bioinformatics significantly in comparison to BnC without the new cutting planes.

However, our experiments show that exact methods may lead to large running times or to a large number of nodes in the BnB/BnC-tree, especially for instances with costs taken uniformly at random from a given set. In particular the BnB algorithm benefits from good start solutions. This motivates the investigation of heuristical approaches for the QTSP. We present several heuristics that are extensions of classical heuristics for the TSP.

Finally, we experimentally compare the algorithms for several real-world instances from bioinformatics and for some randomly generated instances, partially motivated by other applications described above. Most heuristics lead to almost-optimal solutions for the real-world instances, and the Branch-and-Cut algorithm is the fastest exact algorithm for both random and real-world instances. This algorithm is capable of solving large *real-world* instances to proven optimality in reasonable time.

Note that some conclusions from our experiments are rather interesting not only for this specific problem, but possibly also for other combinatorial optimization problems. For example, in our experiments we present example instances, where local search algorithms benefit from relatively poor starting solutions, and other example instances, where the currently leading TSP solver Concorde behaves poorly.

The paper is organized as follows. In Section 2 we describe the motivating problem from bioinformatics that leads to QTSP. In Section 3 we formally introduce the QTSP, QCCP and related problems and study their computational complexity. Furthermore, we develop a polynomial reduction from QTSP to TSP, which is the basis for our first exact algorithm. Our exact and heuristical algorithms for the QTSP are presented in Sections 5 and 6, respectively. We compare the exact methods and the heuristics on several random and real-world instances in Section 7. Finally, we summarize this paper and give suggestions for future research in Section 8.

2. Motivation from Bioinformatics

Gene regulation in higher organisms is accomplished by several cellular processes, two of which are transcription and RNA splicing. In order to better understand these processes, it is desirable to have a good understanding of how transcription factors bind to their DNA binding sites and how the spliceosome binds to RNA splice sites. Many approaches for the computational recognition of transcription factor binding sites or splice sites rely on statistical models, and two of the most promising models for this task are Permuted Markov (PM) models [10] and Permuted Variable Length Markov (PVLM) models [40]. However, finding the optimal PM model or the optimal PVLM model for a given data set is $\mathcal{NP}$-hard. Hence, heuristics for this problem were used in [10,40].

The following paragraphs summarize the key steps that lead from P(VL)M models of order 2 to the QTSP. These paragraphs are not required for understanding the rest of the paper, but they might be helpful for understanding the origin of the QTSP.

Consider a data set x of sequences of length n, and consider a PM model or a PVLM model for modeling these sequences. The central model parameter of a P(VL)M model is a permutation π of positions $1, \ldots, n$.

The log-likelihood of a P(VL)M model of order 1 with given permutation π is given by

$$\log P_1(x|\pi) = b(v_{\pi(1)}) + \sum_{j=2}^{n} c(v_{\pi(j-1)}, v_{\pi(j)}), \tag{1}$$

where b is an n-dimensional vector and c is an $n \times n$ matrix with elements that can be computed analytically from the given data set x [10,40]. Eq. (1) states that the log-likelihood can be written as a sum of a term that depends only on node $v_{\pi(1)}$ and $n-1$ further terms, where the j-th term depends only on the pair of nodes $(v_{\pi(j-1)}, v_{\pi(j)})$.

The problem of finding the permutation π that maximizes the log-likelihood of a P(VL)M model of order 1 can therefore be stated by

$$\hat{\pi} := \arg\max_{\pi} \{\log P_1(x|\pi)\}. \tag{2}$$

This optimization problem can be transformed into a Traveling Salesman Problem (see Section 3 for a formal introduction of the TSP).

Analogously, the log-likelihood of a P(VL)M model of order 2 with given permutation π is given by

$$\log P_2(x|\pi) = b(v_{\pi(1)}, v_{\pi(2)}) + \sum_{j=3}^{n} c(v_{\pi(j-2)}, v_{\pi(j-1)}, v_{\pi(j)}), \tag{3}$$

where b is an $n \times n$ matrix and c is an $n \times n \times n$ tensor with elements that can be computed analytically from the given data set x [10,40]. As above, Eq. (3) states that the log-likelihood can be written as a sum of a term that depends only on the pair of nodes $(v_{\pi(1)}, v_{\pi(2)})$ and $n-2$ further terms, where the j-th term depends only on the triple of nodes $(v_{\pi(j-2)}, v_{\pi(j-1)}, v_{\pi(j)})$.

Consequently, the problem of finding the permutation π that maximizes the log-likelihood of a P(VL)M model of order 2 can be stated by

$$\hat{\pi} := \arg\max_{\pi} \{\log P_2(x|\pi)\}. \tag{4}$$

Analogously, this problem can be transformed into a Traveling Salesman Problem with a quadratic objective function. We refer to Section 3.2 below for a formal description of this transformation.

3. Problem description

3.1. Notations

Let $G = (V, A)$ be a complete directed graph with node set $V = \{1, \ldots, n\}$ and set of arcs $A = V^{(2)} = \{(i,j) \colon i, j \in V, i \neq j\}$. We denote by $V^{(3)} := \{(i,j,k) \colon i,j,k \in V, |\{i,j,k\}| = 3\}$ the set of all 2-arcs associated to the graph G. We often write ij instead of $(i,j) \in V^{(2)}$ as well as ijk instead of $(i,j,k) \in V^{(3)}$. A sequence of nodes $(v_1, v_2, \ldots, v_k)$ is called a *path* if $v_i, v_j \in V, v_i \neq v_j, i, j \in \{1, \ldots, k\}, i \neq j$. A path $(v_1, v_2, \ldots, v_k)$ in G is called *Hamiltonian* if $k = n$. Similarly, a sequence of nodes $(v_1, v_2, \ldots, v_k, v_1)$ is called a *cycle* in G if $v_i, v_j \in V, v_i \neq v_j, i, j \in \{1, \ldots, k\}, i \neq j$. A cycle $(v_1, v_2, \ldots, v_k, v_1)$ is called *Hamiltonian cycle* or *tour* if $k = n$. Throughout the paper we will assume that $n \geq 3$.

Given arc weights $c_l \colon V^{(2)} \to \mathbb{R}$ the total cost of a cycle $C = (v_1, v_2, \ldots, v_k, v_1)$ is

$$c_l(C) := \sum_{i=1}^{k-1} c_l(v_i, v_{i+1}) + c_l(v_k, v_1).$$

The *Traveling Salesman Problem* (TSP) is then

minimize $c_l(T)$,
subject to T is a tour in G.

If c_l is symmetric, *i.e.* if $c_l(u,v) = c_l(v,u)$ for all $u, v \in V, u \neq v$, the problem is also called *Symmetric Traveling Salesman Problem* (STSP), otherwise *Asymmetric Traveling Salesman Problem* (ATSP).

Related to the TSP is the *Weighted Starting Vertex Hamiltonian Path Problem* (SVHPP). Instead of a tour, the SVHPP asks for a Hamiltonian path $P = (v_1, \ldots, v_n)$ with minimal costs, *i.e.* with

$$c_l(P) := \sum_{i=1}^{n-1} c_l(v_i, v_{i+1}),$$

the SVHPP reads

minimize $c_l(P)$,

subject to P is a Hamiltonian path in G.

Another related problem is the *Cycle Cover Problem* (CCP), which asks for a set of cycles $K = \{C_1, \dots, C_k\}$ such that each node is contained in exactly one cycle and the sum of the arc weights contained in these cycles is minimized (note that the CCP is equivalent to a linear assignment problem).

The *Quadratic Traveling Salesman Problem* (QTSP) differs from the TSP in that the costs of a tour do not depend only on each two but on each three nodes contained in succession in the tour. Formally, given a weight function $c_q\colon V^{(3)} \to \mathbb{R}$, the costs of a cycle $C = (v_1, \dots, v_k, v_1)$ are

$$c_q(C) := \sum_{i=1}^{k-2} c_q(v_i, v_{i+1}, v_{i+2}), +c_q(v_{k-1}, v_k, v_1) + c_q(v_k, v_1, v_2),$$

and the QTSP reads

minimize $c_q(T)$,

subject to T is a tour in G.

If c_q is symmetric, *i.e.* if $c_q(u, v, w) = c_q(w, v, u)$ for all $u, v, w \in V$, $|\{u, v, w\}| = 3$, the problem is also called *Symmetric Quadratic Traveling Salesman Problem* (SQTSP), otherwise *Asymmetric Quadratic Traveling Salesman Problem* (AQTSP).

Similarly to the standard case, the *Quadratic Starting Vertex Hamiltonian Path Problem* (QSVHPP) asks for a Hamiltonian path with minimal costs w.r.t. c_q, and the *Quadratic Cycle Cover Problem* (QCCP) asks for a cycle cover with minimal costs w.r.t. c_q.

Remark 3.1. SVHPP can be easily transformed to TSP, as well as QSVHPP to QTSP by adding an additional artificial node.

3.2. Transformation of instances from Bioinformatics to a QTSP

We describe shortly how the optimization problem (4) can be transformed to a QTSP. Let n be the length of the DNA sequences. Then we define the complete graph $G = (V, A)$ with set of nodes $V = \{0, 1, \dots, n\}$. A permutation $\pi\colon \{1, \dots, n\} \to \{1, \dots, n\}$ corresponds to a tour $(0, \pi(1), \dots, \pi(n), 0)$ in G. The artificial node 0 is used to model the first summand in (3). Note that all values $b(v_{\pi(1)}, v_{\pi(2)})$ and $c(v_{\pi(j-2)}, v_{\pi(j-1)}, v_{\pi(j)})$, $j \in \{3, \dots, n\}$, are nonpositive because they are logarithms of probabilities. So with

$$c_q(i, j, k) := \begin{cases} -b(j, k), & i = 0, \\ 0, & j = 0 \vee k = 0, \\ -c(i, j, k), & \text{otherwise}, \end{cases}$$

we get

$$\min\left\{c_q(T)\colon T \text{ is a tour in } G\right\} = -\max\left\{\log P_2(x|\pi)\colon \pi \text{ is a permutation of } 1, \dots, n\right\}.$$

This is a QTSP as defined above.

4. Basic results

4.1. Complexity results

The (decision variants of the) Hamiltonian path problem and the traveling salesman problem are well-known to be $\mathcal{NP}$-complete [14]. As QTSP and QSVHPP are generalizations of TSP and SVHPP, respectively, both problems are $\mathcal{NP}$-complete, too. This can be shown by defining $c_q(u, v, w) := c_l(u, v)$ for all $(u, v, w) \in V^{(3)}$.

However, the situation is different for CCP and QCCP. Whereas CCP is solvable in polynomial time by the Hungarian method [28], QCCP is also $\mathcal{NP}$-complete. The $\mathcal{NP}$-completeness of QCCP easily follows from the $\mathcal{NP}$-completeness of the more specific *Angular-Metric Cycle Cover Problem* [1], which is the problem of minimizing the total angle change of a set of cycles for a set of points in the Euclidean space, where the weight of a cycle is the sum of the angle changes at its points. In the Appendix we provide a different proof of the $\mathcal{NP}$-completeness of QCCP, which uses a much simpler construction.

Theorem 4.1. *The decision problem of QCCP is $\mathcal{NP}$-complete.*

Note that this theorem is an important basis for creating and modifying a cycle patching heuristic for the QTSP in Section 6.

A similar result like Theorem 4.1 holds for an extension of the TSP, namely the *Generalized Traveling Salesman Problem* (GTSP) which will be introduced in the next section (see [22] for a proof for this result).

4.2. Polynomial reduction from QTSP to TSP

The TSP is one of the best studied combinatorial optimization problems, and there are very sophisticated software packages for the solution of TSP, *e.g.*, the well-known STSP-solver CONCORDE [7], which has successfully been used to solve very large TSP-instances to optimality. Unfortunately, these packages cannot be used to solve QTSP-instances directly. However, because QTSP and TSP are both $\mathcal{NP}$-complete problems, we know that a QTSP-instance can be transformed into an instance of TSP in polynomial time. In the following we present one such transformation. This transformation of QTSP to ATSP, and via the transformation of Jonker and Volgenant [28] to STSP allows us to use the CONCORDE-solver for the solution of QTSP. We will present computational results in Section 7.

As substep from QTSP to TSP we first transform QTSP into a so called *Generalized Traveling Salesman Problem* (GTSP) [12]. The asymmetric GTSP is formulated as follows. We are given a complete arc-weighted directed graph $\tilde{G} = (\tilde{V}, \tilde{A})$ with node set $\tilde{V}$, set of arcs $\tilde{A}$ and arc-weights $c: \tilde{V}^{(2)} \to \mathbb{R}$. The node set $\tilde{V}$ can be partitioned into k node sets $V_1, \ldots, V_k \subset \tilde{V}$, $k \geq 3$, so called *clusters*, such that $\tilde{V} = V_1 \dot{\cup} \ldots \dot{\cup} V_k$, $V_i \cap V_j = \emptyset$, $i, j = 1, \ldots, k$, $i \neq j$. The task is to determine an optimal directed cycle in $\tilde{G}$ such that each cluster V_i, $i = 1, \ldots, k$, is visited at least once. We will consider here the slightly different variant E-GTSP that asks for a cycle that visits exactly one node of each cluster.

There are several transformations known in the literature that allow to solve E-GTSP (GTSP) as an ATSP or even STSP, see, *e.g.*, [9]. Some of them even do not enlarge the number of nodes [35,6] in the asymmetric case.

We will transform a QTSP instance in three steps to an STSP instance to be solved with CONCORDE. In a first step, we transform a QTSP instance to an E-GTSP instance using the following Theorem 4.2. Second, the E-GTSP instance is transformed to an ATSP instance using the construction of Behzad and Modarres [6]. Finally, the ATSP instance is transformed to an STSP instance using the approach of Jonker and Volgenant [28]. The final STSP instance will have $2n(n-1)$ nodes.

Now we show how to transform QTSP to E-GTSP.

Theorem 4.2. *Given an instance of AQTSP on a directed complete graph $G = (V, A)$ with weights $c_q: V^{(3)} \to \mathbb{R}$, consider the E-GTSP instance on a directed graph $\tilde{G} = (\tilde{V}, \tilde{A})$ with $\tilde{V} := \{uv: (u, v) \in V^{(2)}\}$ and $\tilde{A} := \{(uv, u'v') \in \tilde{V} \times \tilde{V}: uv \neq u'v'\}$ and arc weights $\tilde{c}: \tilde{A} \to \mathbb{R}$ with*

$$\tilde{c}(uv, u'v') := \begin{cases} c_q(u, v, v'), & \textit{if } v = u', u \neq v' \\ 2M, & \textit{otherwise,} \end{cases}$$

where $M := \sum_{(u,v,w) \in V^{(3)}} |c_q(u, v, w)| + 1$. Then both problems are equivalent.

Proof. Each solution of the constructed E-GTSP instance with objective value less than M, in particular each optimal solution, visits each of the clusters exactly once and contains exactly one node of each cluster. Furthermore, such cycles contain only arcs $(uv, u'v')$ with $v = u'$ and $u \neq v'$ because otherwise the costs of the cycle would be at least M. So we get a one-to-one correspondence between optimal solutions of the constructed E-GTSP-instance and the QTSP-instance. Let $C = (u_1v_1, u_2v_2, \ldots, u_nv_n, u_1v_1)$ be an optimal solution of the E-GTSP-instance. Then we know by the considerations above that $v_i = u_{i+1}$, $i \in \{1, \ldots, n-1\}$, $v_n = u_1$ and $u_i \neq u_j$ for all $i, j \in \{1, \ldots, n\}$, $i \neq j$. Hence, the cycle $(u_1, \ldots, u_n, u_1)$ is an optimal solution of the QTSP-instance with the same objective value. The other way round we can specify an optimal solution of the E-GTSP-instance from an optimal solution of QTSP. □

5. Exact algorithms for QTSP

In this section we present three exact algorithms for solving the QTSP. The first one, presented in Section 5.1, uses the transformation of Section 4.2 to transform the QTSP instance into an STSP instance to be solved by a standard solver of the STSP. The second approach in Section 5.2 is a Branch-and-Bound algorithm based on combinatorial lower bounds. Third, we present a Branch-and-Cut algorithm in Section 5.3. In our experiments we used two variants of the Branch-and-Cut algorithm, one that only separates the subtour elimination constraints, and one that separates several additional cutting planes to be presented in Section 5.3.

5.1. Solving as STSP

In Section 4.2 we have presented one possible approach for transforming QTSP to an E-GTSP and afterwards to an ATSP or STSP. Because the TSP-solver CONCORDE can only handle symmetric instances, we describe below the coefficients in the objective function of the final STSP (for using an ATSP-solver, the last step can be omitted). We assume, w.l.o.g., that all weights are nonnegative (otherwise we can add a sufficiently large constant to all weights). We only give the final construction that can be obtained by applying the constructions of Theorem 4.2, [6] and [27] in order.

Given a graph $G = (V, A)$ with $V = \{1, \ldots, n\}$ and weights $c_q: V^{(3)} \to \mathbb{R}_+$, the final graph is $\bar{G} = (\bar{V}, \bar{A})$ with

$$\bar{V} := \{ij^+, ij^-: ij \in V^{(2)}\} \quad \text{and} \quad \bar{A} := \{\{u, v\}: u, v \in \bar{V}, u \neq v\}.$$

For each node $i \in V$ we arrange the nodes $j \in V \setminus \{i\}$ in a cycle and denote the successor of j on the cycle for i by $s(i,j)$, *e.g.*, for $i = 1$ and cycle $(2, \dots, n, 2)$ it is $s(1,j) = j + 1$ for $j < n$ and $s(1,n) = 2$. The objective function is given by

$$\bar{c}(ij^-, kl^-) = \bar{c}(ij^+, kl^+) := \infty,$$

$$\bar{c}(ij^+, kl^-) = \bar{c}(kl^-, ij^+) := \begin{cases} -n \cdot M, & ij = kl, \\ 0, & i = k, l = s(i,j), \\ c_q(i,k,l) + M, & i \neq l, s(i,j) = k, i \neq k, \\ \infty, & \text{otherwise}, \end{cases}$$

where M is a sufficiently large constant, *e.g.*, $M = c_q(T)$ for some arbitrary tour T in G. Depending on the solver, edges $(u,v) \in \bar{A}$ with $\bar{c}(u,v) = \infty$ can be omitted from the graph or one can use a sufficiently large constant, *e.g.*, $M \cdot (n+1)^2$. Note, in order to solve these instances by some standard TSP solvers it might be necessary to increase all coefficients by $n \cdot M$ such that the final coefficients are nonnegative.

5.2. Branch-and-bound algorithm

The Branch-and-Bound algorithm (BnB) traverses in the worst case all possible tours and computes the tour with minimal costs. To avoid traversing all tours, it computes (local) lower bounds and upper bounds by traversing and analyzing subpaths of all possible tours.

First, we start with an arbitrary QTSP heuristic in order to compute a good upper bound. Each time a new subpath is considered, a lower bound for a QTSP solution containing this subpath is computed. In our BnB approach we use as lower bound the solution of the CCP problem w.r.t. the objective function

$$c_l(u,v) := \min_{w \in V \setminus \{u,v\}} c_q(u,v,w)$$

(see also the description of the AP heuristic in Section 6). If the lower bound is greater than or equal to the current upper bound, *i.e.*, the best currently known solution, we can prune this branch. The upper bound is updated if a tour with smaller costs is found. All tours are started with a fixed node v_1, which is chosen in such a way that the sum over all values $c_q(v_1, x, y)$ with $x \neq v_1, y \neq v_1, x \neq y$ is maximal. This choice is used because we expect a higher amount of pruning if the lower bounds in the first steps are rather large.

5.3. Branch-and-cut algorithm

5.3.1. Integer-programming model

The most successful approach for the TSP is using an Integer Programming (IP) formulation based on arc variables. We assign to each arc $a \in A$ a binary variable $x_a \in \{0,1\}$ with the interpretation $x_a = 1$ if and only if the arc a is contained in the tour and zero otherwise. The IP formulation of Dantzig et al. [8] reads

$$\text{minimize} \sum_{(u,v) \in A} c_l(u,v) \cdot x_{(u,v)}$$

$$\text{subject to} \sum_{v \in V:(u,v) \in A} x_{(u,v)} = \sum_{v \in V:(v,u) \in A} x_{(v,u)} = 1, \quad u \in V, \tag{5}$$

$$\sum_{(u,v) \in A: u,v \in S} x_{(u,v)} \leq |S| - 1, \quad \emptyset \neq S \subsetneq V, \tag{6}$$

$$x \in \{0,1\}^A. \tag{7}$$

Constraints (5) ensure that each node is entered and left exactly once by the tour. Conditions (6) are the well-known *Subtour Elimination Constraints* (SEC), forbidding cycles of length less than n. Finally, constraint (7) ensures the integrality of the variables. Note, we get a formulation for CCP by omitting the SEC (6).

Concerning QTSP, the cost coefficient $c_q(u,v,w)$ is counted for a tour if and only if the two corresponding arcs (u,v) and (v,w) are contained in the tour. Hence, we can formulate QTSP as an integer program with quadratic objective function

$$\text{minimize} \sum_{(u,v,w) \in V^{(3)}} c_q(u,v,w) \cdot x_{(u,v)} \cdot x_{(v,w)}$$

subject to (5), (6), (7).

By introducing a new variable $y_{(u,v,w)}$ for each product $x_{(u,v)} \cdot x_{(v,w)}$, one gets a linear integer programming formulation for QTSP

$$\text{minimize} \sum_{(u,v,w) \in V^{(3)}} c_q(u,v,w) \cdot y_{(u,v,w)}$$

subject to (5), (6), (7),

$$x_{(u,v)} = \sum_{\substack{w \in V: \\ (u,v,w) \in V^{(3)}}} y_{(u,v,w)} = \sum_{\substack{w \in V: \\ (w,u,v) \in V^{(3)}}} y_{(w,u,v)}, \quad (u,v) \in A, \tag{8}$$

$$y \in [0,1]^{V^{(3)}}. \tag{9}$$

Eqs. (8) can be interpreted in the following way. If an arc $(u, v) \in A$ is contained in the tour, there has to be a node $w \in V$ so that the path (u, v, w) is part of the tour and a node $w' \in V$ so that the path (w', u, v) is part of the tour. Similar constraints occur in a linearization of the *Quadratic Assignment Problem* (QAP), see, *e.g.*, [13]. The following result shows that the model above is indeed a formulation for QTSP.

Lemma 5.1. *A vector $(x, y) \in \{0, 1\}^A \times [0, 1]^{V^{(3)}}$ satisfies all constraints (5)–(9) if and only if the x-variables correspond to a tour and there holds $x_{(u,v)} \cdot x_{(v,w)} = y_{(u,v,w)}$ for all $(u, v, w) \in V^{(3)}$.*

Proof. The x-variables satisfy (5)–(7) if and only if x corresponds to a tour because these constraints are a formulation of TSP. It remains to prove $x_{(u,v)} \cdot x_{(v,w)} = y_{(u,v,w)}$ for all $(u, v, w) \in V^{(3)}$.

First, w.l.o.g., let $x_{(u,v)} = 0$. Then by (8) and (9) it follows $y_{(u,v,w)} = y_{(w,u,v)} = 0$ for all $w \in V \setminus \{u, v\}$. It remains to consider the case $x_{(u,v)} = x_{(v,w)} = 1$. Assume $y_{(u,v,w)} < 1$. Then there exists $w' \neq w$ with $y_{(u,v,w')} > 0$ by (8), and this implies $x_{(v,w')} = 1$ by (8). This is a contradiction to (5). Thus $y_{(u,v,w)} = 1$. □

Remark 5.2. We get an integer programming formulation for the QCCP by the constraints (5) and (7)–(9).

5.3.2. The algorithm and additional cutting planes

An alternative to using combinatorial bounds in a BnB approach is the utilization of linear programming (LP) relaxations. An LP-relaxation can be obtained from an IP-formulation by replacing integrality constraints by simple box constraints, *i.e.* in the case of the TSP the LP-relaxation reads

$$\text{minimize} \sum_{(u,v) \in A} c_l(u,v) \cdot x_{(u,v)}$$

subject to (5), (6),

$$x \in [0,1]^A, \tag{10}$$

which is a standard linear program, which can be solved efficiently. However, the LP-relaxation still contains an exponential number of constraints (6). Thus, it is practically impossible to add all constraints to the model at once. Fortunately, it is possible to combine linear programming with a so called cutting-plane approach, which starts with only a small number of constraints and adds additional constraints during the solution process if they are violated. It is well-known that cutting plane approaches can be used to solve an LP in polynomial time by the ellipsoid method (see, *e.g.*, [39]), if the corresponding separation problems can be solved in polynomial time [20]. In addition to the necessary constraints for a formulation, one usually separates further cutting planes that tighten the LP-relaxation in order to get even stronger bounds. Combining cutting planes with a BnB approach is called *Branch-and-Cut* (BnC).

In fact, the BnC approach is the most successful approach for exactly solving the STSP [8,19,36]. For instance, the BnC based TSP solver Concorde [3,7] has solved a TSP instance of 85,900 cities [4]. This is possible because the subtour elimination constraints (6) can indeed be separated in polynomial time [25], but Concorde also uses several further cutting planes.

We apply similar techniques to the QTSP. By replacing the integrality constraints by box constraints we get the LP-relaxation for the QTSP

$$\text{minimize} \sum_{(u,v,w) \in V^{(3)}} c_q(u,v,w) \cdot y_{(u,v,w)}$$

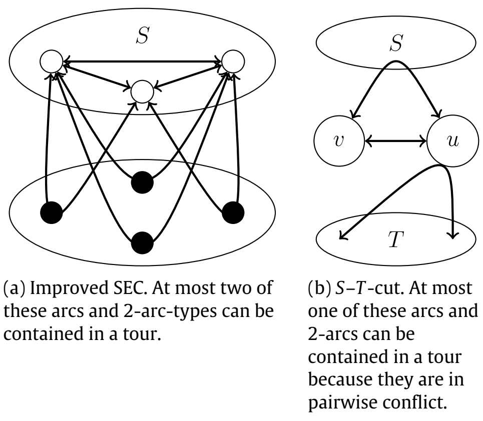
subject to (5), (6), (8), (9), (10).

In order to get an exact algorithm for the QTSP, it is sufficient to separate only the standard subtour elimination constraints (6). In our numerical experiments we also tried this basic BnC approach, which we will denote by BnC-S. However, we also improved the LP-relaxation by additional cutting planes to be presented next. The respective algorithm is called BnC-E.

1. The SEC can be strengthened as follows:

$$\sum_{\substack{(u,v) \in A: \\ u,v \in S}} x_{(u,v)} + \sum_{\substack{(u,v,w) \in V^{(3)}: \\ u,w \in S, v \notin S}} y_{(u,v,w)} \leq |S| - 1, \quad S \subset V, 2 \leq |S| < \frac{n}{2}, \tag{11}$$

   where the subset S is required to have cardinality smaller than $n/2$. This strengthening is correct, as not only the direct connections between two nodes are counted, but also the connections with one node between them (see Fig. 1(a)). As



(a) Improved SEC. At most two of these arcs and 2-arc-types can be contained in a tour.

(b) S–T-cut. At most one of these arcs and 2-arcs can be contained in a tour because they are in pairwise conflict.

Fig. 1. Visualization of certain cutting planes that are used to strengthen the LP-relaxation.

no easy polynomial-time separation algorithm exists for these cuts, we heuristically separate them if one of the standard SEC is violated. Furthermore we explicitly separate all inequalities for $|S| = 2$. (Recently, it has been shown in [11] that determining a maximally violated constraint of type (11) is $\mathcal{NP}$-complete.)

2. The following cuts are feasible for $|V| \geq 4$ and forbid all subtours containing only 3 nodes:

$$y_{(u,v,w)} + y_{(w,u,v)} \leq x_{(u,v)}, \quad u, v, w \in V, |\{u, v, w\}| = 3. \tag{12}$$

Because only $\mathcal{O}(n^3)$ such inequalities exist, they can be separated in polynomial time.

3. Let $u, v \in V, u \neq v, S \subset V \setminus \{u, v\}, T := V \setminus (S \cup \{u, v\})$ and $|V| \geq 5$. Then only one of the following variables can be set to 1, as otherwise we would have a cycle or a forbidden T-structure, *i.e.*, a node with degree three (see Fig. 1(b)).
   - $x_{(u,v)}$ or $x_{(v,u)}$,
   - $y_{(u,w,v)}$ or $y_{(v,w,u)}$, $w \in S$,
   - $y_{(p,u,q)}$, $p, q \in T, p \neq q$.

   This leads to the S–T-cuts of the form

$$x_{(u,v)} + x_{(v,u)} + \sum_{w \in S}(y_{(u,w,v)} + y_{(v,w,u)}) + \sum_{p,q \in T: p \neq q} y_{(p,u,q)} \leq 1. \tag{13}$$

Although there is an exponential number of these inequalities they can be separated in polynomial time, see the following remark.

Remark 5.3. Given a fractional solution $(\bar{x}, \bar{y})$ of a relaxation of AQTSP the problem of determining sets S, T maximizing the sum in (13) for fixed $u, v \in V, u \neq v$, can be formulated as an integer program whose corresponding constraint matrix is totally unimodular. We introduce binary variables $s_k \in \{0, 1\}$ for each node $k \in V \setminus \{u, v\}$ and $t_{kl} \in \{0, 1\}$ for each $kl \in (V^{\{2\}} \setminus \{\{u', v'\} \in V^{\{2\}} : \{u', v'\} \cap \{u, v\} \neq \emptyset\}) =: \hat{V}^{\{2\}}$ with the interpretation

$$s_k = \begin{cases} 1, & k \in S, \\ 0, & k \notin S, \end{cases} \quad \text{and} \quad t_{kl} = \begin{cases} 1, & \{k, l\} \subset T, \\ 0, & \{k, l\} \not\subset T, \end{cases}$$

and coefficients $c_{s_k} = \bar{y}_{ukv} + \bar{y}_{vku}, k \in V \setminus \{u, v\}$, and $c_{t_{kl}} = \bar{y}_{kul} + \bar{y}_{luk}, kl \in \hat{V}^{\{2\}}$, in the objective function. The inequalities

$$s_k + t_{kl} \leq 1 \quad \text{and} \quad s_l + t_{kl} \leq 1, \quad kl \in \hat{V}^{\{2\}},$$

forbid that a node $u' \in V \setminus \{u, v\}$ is contained in both sets S and T. Regarding the constraint matrix $\tilde{A} \in \{0, 1\}^{2\binom{n-2}{2} \times (n-2) + \binom{n-2}{2}}$ each row has exactly two nonzero entries. Applying the Theorem of Heller and Tompkins, see, *e.g.*, Corollary 2.8 in [34], to $\tilde{A}^T$ with the rows corresponding to variables $s_k, k \in V \setminus \{u, v\}$, in one class and the rows corresponding to variables $t_{kl}, kl \in \hat{V}^{\{2\}}$, in a second class proves that $\tilde{A}$ is totally unimodular. So we can solve the separation problem for inequalities (13) using linear programming methods. Recently, it has been shown by [11] that the separation problem can be reduced to a *maximum weighted independent set problem in bipartite graphs*.

6. Heuristics for QTSP

In our experiments it turned out that for certain instance classes the exact methods presented in the previous section are quite time consuming for large n. Furthermore, good start solutions are beneficial for the BnB algorithm. This motivates the investigation of heuristical approaches. In this section we present several simple heuristics, all of which are adaptions of classical heuristics for the TSP.

Cheapest-Insert Heuristic (CI). This is a generalization of an ATSP heuristic [37]. We start with an arc $(v_1, v_2) \in A$ considered as a cycle and choose this arc so that the term

$$\min_{x\in V} c_q(x, v_1, v_2) + \min_{x\in V} c_q(v_1, v_2, x)$$

is minimal. Note that the natural starting point, namely starting with a pair of arcs (v_1, v_2) and (v_2, v_3), so that $c_q(v_1, v_2, v_3)$ is minimal over all pairwise distinct triples, would lead to a bad tour, if $c_q(v_2, v_3, x)$ is large for all $x \in V \setminus \{v_2, v_3\}$ or if $c_q(x, v_1, v_2)$ is large for all $x \in V \setminus \{v_1, v_2\}$. The new nodes are iteratively included in the cycle in a greedy manner, so that in each step the new cycle is cost minimal. The heuristic stops when the cycle is a tour.

Nearest-Neighbor Heuristic (NN). This is also a generalization of an ATSP heuristic [37]. Given a path $P_{k-1} = (v_1, \ldots, v_k)$ we append a node $v_{k+1} \in V \setminus \{v_1, \ldots, v_k\}$ so that $c_q(v_{k-1}, v_k, v_{k+1})$ is minimal. The arc $(v_1, v_2) \in A$ for the first iteration is chosen so that

$$\frac{1}{n-2} \cdot \left(\sum_{x\in V} c_q(x, v_1, v_2)\right) + \min_{x\in V} c_q(v_1, v_2, x)$$

is minimal in order to respect the predecessor of v_1 to be chosen in the last step.

Two-Directional-Nearest-Neighbor Heuristic (*2NN*). A slight variation of the NN heuristic is the *Two Directional Nearest-Neighbor Heuristic*, which differs in two aspects from NN. First, the heuristic considers both directions to find the next neighbor. One possible criterion for choosing the next node would be to use the minimal cost neighbor over all new nodes and over both directions. Note, however, that the tour has to be closed anyway, so that *both* directions have to be used now or at a later step. Thus for a given path $(v_1, v_2, \ldots, v_i)$, the cost values $c_q(v_{i-1}, v_i, x)$ and $c_q(y, v_1, v_2)$ for a cost minimal successor x and predecessor y, respectively, are less important than the difference to the second smallest values in both directions. For both directions, this value can be viewed as an *upper tolerance* of the problem of finding a cost minimal neighbor node (for an overview over the theory of tolerances see [17]). The 2NN heuristic chooses the direction for which the upper tolerance value is larger because not using the cost minimal neighbor node would cause a larger jump of the costs at a later step. A similar idea was used for a tolerance based version [16] of the ATSP Greedy Heuristic [15] and a tolerance based version [18] of the ATSP Contract-or-Patch Heuristic [15].

Assignment-Patching Heuristic (AP). A well-known technique for ATSP is the patching technique. It starts from a feasible CCP solution and then, step by step, patches two cycles together, until there is only one cycle. Because the optimal CCP solution can be computed efficiently and the solution value is often a good lower bound for an optimal ATSP solution value, it is a good starting point for patching. Karp and Steele [29] suggested for each step to patch the two cycles containing the highest number of nodes. Two cycles C_1 and C_2 are patched by replacing two arcs $e_1 = (v_1, w_1) \in C_1$ and $e_2 = (v_2, w_2) \in C_2$ by the two other arcs (v_1, w_2) and (v_2, w_1) so that the resulting cycle has minimal costs. We denote this patching technique by KSP. An analogous heuristic can be used for the QTSP. However, by Theorem 4.1, QCCP is $\mathcal{NP}$-hard (in contrast to CCP). Hence, we determine an approximate solution of QCCP by computing a CCP solution using the following objective function

$$c_l(u, v) := \min_{w\in V\setminus\{u,v\}} c_q(u, v, w).$$

Then we patch the cycles of this CCP solution using KSP and get a feasible QTSP solution. We call the approach *Assignment-Patching Heuristic* (AP).

Nearest-Neighbor-Patching Heuristic (NNP). This is a combination of NN and AP heuristics. If closing the current path in the NN heuristic would lead to a good subtour, the cycle is closed and NN starts again on the remaining nodes. This leads to a cycle cover, which is patched afterwards using AP.

The main decision is when to close the current path. Experiments (not presented in this work) have shown that, given the current path $P = (v_1, \ldots, v_i)$, closing the path to a subtour if $c_q(v_{i-1}, v_i, v_1) + c_q(v_i, v_1, v_2) \le 2 \cdot \sum_{j=1}^{i-2} c_q(v_j, v_{j+1}, v_{j+2})$ is a good choice. Because all cycles should contain at least three nodes and the rest of the graph has also to be divided into cycles, we have to ensure that the path contains at least three nodes and at least three nodes remain in the rest.

Two Directional Nearest-Neighbor-Patching Heuristic (*2NNP*). Analogously to NNP, this heuristic is a combination of 2NN and AP.

Greedy Heuristic (GR). This is again a generalization of an ATSP heuristic [15], which is based on a contraction procedure. Given a graph $G = (V, A)$ the greedy heuristic chooses an arc $a = (v_1, v_2)$ to be contained in the tour and constructs a new graph $\tilde{G} = (\tilde{V}, \tilde{A})$ by contracting a to a new node v_a. The heuristic computes recursively a tour in $\tilde{G}$ w.r.t. the objective function

$$\tilde{c_q}(u, v, w) := \begin{cases} c_q(v_2, v, w), & v_a = u, \\ c_q(u, v, v_1), & v_a = w, \\ c_q(u, v_1, v_2) + c_q(v_1, v_2, w), & v_a = v, \\ c_q(u, v, w), & \text{otherwise.} \end{cases}$$

If $|\tilde{V}| = 3$, then the heuristic simply returns the cheaper one of the two possible tours. The final tour is obtained by expanding the selected arcs in reverse order. The GR heuristic starts with contracting a "good" arc. It is reasonable to choose this arc in the same way as in the CI heuristic.

k-OPT Heuristic. All previously presented heuristics explicitly construct tours from scratch. In contrast, the k-OPT heuristic modifies an existing tour in order to reduce its length. It is completely analogous to the well-known k-OPT heuristic for the TSP [31]: For k pairwise different arcs, the heuristic replaces these arcs by k other arcs so that the weight of the resulting tour is minimal. This is repeated until no further improvement can be achieved. We investigate only the case $k \leq 3$ in our experiments. Note that the k-OPT heuristic can be used to improve the tour found by any other heuristic.

Remark 6.1. The main reason for the differentiation between the ATSP and the STSP is the fact that for the STSP specific STSP algorithms are used instead of general ATSP algorithms. This holds for both, heuristics (compare Helsgaun's LKH Heuristic [24]) and exact algorithms (compare the solver Concorde [3,7]). Note that both, LKH and Concorde, can also be applied to asymmetric instances by the 2-point reduction method, see [21, Chapter 2], [28].

The exact methods presented here work for the asymmetric case, but clearly, they can also be applied to the symmetric case. However, for the symmetric case optimizations are possible again. For example, one can halve the number of variables of the IP. Furthermore, for the STSP in the k-OPT improvement heuristic more rearrangements of the tours are allowed, as the direction of the tour parts does not have to be considered. We have used these optimizations in the experiments regarding symmetric instances.

7. Experimental study

We implemented all algorithms in C++, where we used the following subroutines for the QTSP algorithms. For the TSP-R Algorithm we used the TSP solver Concorde [3,7]. For the BnB Algorithm we used the CCP solver implemented by Jonker and Volgenant [28], which is based on the Hungarian method. The implementation of the BnC Algorithm is based on the BnC-solver Cplex [26], which is extended by problem-specific cutting planes. For the subtour elimination constraints we employ the minimum cut algorithms in the software package Lemon [30].

All experiments were carried out on a PC with an Intel® Core™ i7 CPU 920, 2.67 GHz, 12 GB RAM. As test instances we chose six classes of random instances—two classes of asymmetric instances and four of symmetric instances. Furthermore, we tested several asymmetric real-world instances from bioinformatics in order to find an optimal PM model of order 2 for certain splice and donor sites.

We investigate six classes of random instances. Whereas the first four classes are natural extensions of classes of random TSP instances used in [15], the last two classes are symmetric (extended) angular-metric instances. The six classes can be described as follows.

- Asymmetric Class 1:
  Each entry $c_q(i,j,k)$, $(i,j,k) \in V^{(3)}$, is chosen uniformly at random as an integer from $\{0, 1, \ldots, 10\,000\}$.
- Asymmetric Class 2:
  Each entry $c_q(i,j,k)$, $(i,j,k) \in V^{(3)}$, is chosen uniformly at random as an integer from $\{0, 1, \ldots, i \cdot j \cdot k - 1\}$.
- Symmetric Class 1:
  Each entry $c_q(i,j,k) = c_q(k,j,i)$, $(i,j,k) \in V^{(3)}, i < k$, is chosen uniformly at random as an integer from $\{0, 1, \ldots, 10\,000\}$.
- Symmetric Class 2:
  Each entry $c_q(i,j,k) = c_q(k,j,i)$, $(i,j,k) \in V^{(3)}$, $i < k$, is chosen uniformly at random as an integer from $\{0, 1, \ldots, i \cdot j \cdot k - 1\}$.
- Symmetric Class 3:
  For constructing random classical angular-metric SQTSP-instances, n points $v^1, \ldots, v^n$ are chosen uniformly at random out of $[0, 500]^2 \cap \mathbb{N}_0^2$. The coefficients are calculated according to
  $$c_q^{\angle}(i,j,k) := \left\lfloor 1000 \cdot \left(\arccos \left(\left(\frac{v^j - v^i}{\|v^j - v^i\|} \right)^T \left(\frac{v^k - v^j}{\|v^k - v^j\|} \right) \right) \right) \right\rfloor .$$
- Symmetric Class 4:
  For constructing random extended angular-metric SQTSP instances, where the angle is weighted against the length of the tour (see also the two-step approaches for solving the TSP for Dubins vehicle in [33,32]), n points $v^1, \ldots, v^n$ are chosen uniformly at random out of $[0, 500]^2 \cap \mathbb{N}_0^2$. The coefficients are calculated according to
  $$c_q^{\angle,D}(i,j,k) := \left\lfloor 10 \cdot \left(\tfrac{1}{2}\|v^i - v^j\| + \tfrac{1}{2}\|v^j - v^k\| + \varrho \cdot c_q^{\angle}(i,j,k) \right) \right\rfloor .$$
  As in [33] we use a turn radius of $\varrho = 40$.

Regarding the random instances, we computed the average over 10 instances for the exact algorithms and over 100 instances for the heuristics.

7.1. Comparison of exact algorithms

In this section we compare the running times of all introduced exact algorithms, namely the TSP algorithm TSP-R, the BnB algorithm and the BnC algorithms BnC-S and BnC-E. The results can be found in Figs. 2–4. In the following we describe for each algorithm the main observations and give a short analysis.

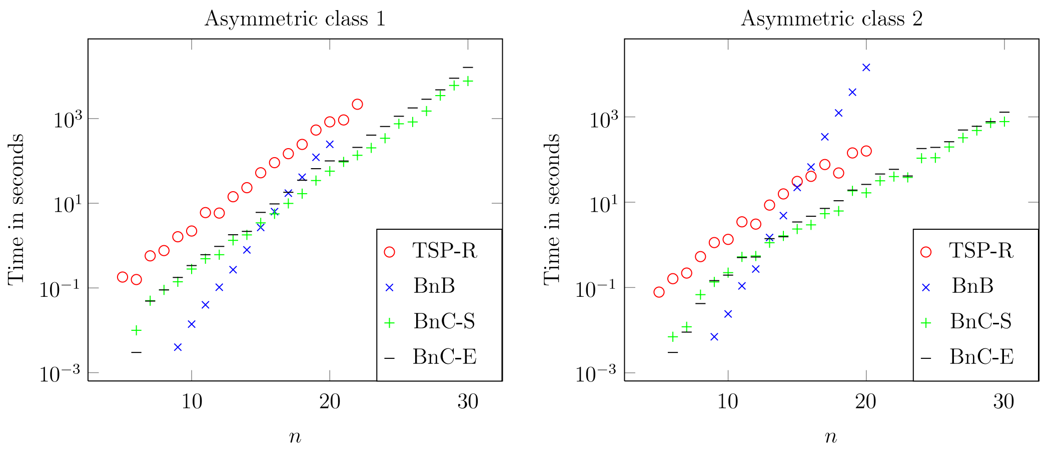


Fig. 2. Time for the exact algorithms applied to the asymmetric random classes.

- **TSP-R:** For all classes, the TSP-R algorithm is able to solve instances up to $n = 20$ in reasonable time. For larger n, CONCORDE often failed for these instances as the coefficients were too large. The branch-and-cut approach is typically faster, but TSP-R is better than the BnB algorithm for instances in asymmetric class 2, in symmetric classes 2 and 3 as well as in real-world class 1 if n is large enough. TSP-R is the worst algorithm for asymmetric class 1 and symmetric class 4. Note that the quality of TSP-R corresponds to the performance of CONCORDE applied to the TSP instance which has been constructed by the given QTSP instance. For a QTSP instance with 15 nodes this leads to a TSP instance with $2 \cdot 15 \cdot 14 = 420$ nodes (see Section 5.1). At first sight, this performance of CONCORDE in comparison to BnC is surprising, as CONCORDE has solved even a TSP instance with 85,900 nodes [4]. We suppose that the structure of the considered TSP instances, which are neither Euclidean instances nor random instances, is rather bad for the application of CONCORDE, as it contains a large number of solutions with objective value close to the optimal solution value. Furthermore, because of the large constants during the transformation, many coefficients of the TSP instances are rather large, which could also cause numerical problems for CONCORDE.
- **BnB:** If the number of nodes is small, this algorithm is the fastest one, but for a large number of nodes BnC clearly beats it. Especially for asymmetric class 2, symmetric classes 2 and 3 as well as for real-world class 1 it is the worst algorithm.
- **BnC:** Comparing both BnC versions, namely the basic version BnC-S and the extended version BnC-E, the behavior of these variants strongly depends on the instance class. For random instances from the asymmetric classes and the symmetric classes 1 and 2, BnC-S is the best algorithm and often faster than BnC-E. On the other hand, for instances of the symmetric classes 3 and 4, BnC-E runs faster, for the real-world instances even with some orders of magnitude. *In particular BnC-E is able to solve all real-world instances up to dimension 100 in about ten minutes (see* Fig. 4*)*. This time difference can be explained as follows. There are only few nodes in the branch-and-cut tree for BnC-E, and often the solution is even found in the root node. So for these instances the newly derived cutting planes are very effective. For some of the random instances the value of the first linear relaxation is mostly far away from the optimal solution value leading to many branching steps. In these branching steps, the computation of the cutting planes is not able to do an essential reduction of the branching tree, but it costs much time.

7.2. *Comparison of heuristics*

An experimental study of ATSP heuristics is given in [21, Chapter 10]. In this section we make a similar study for the QTSP. More precisely, we compare all considered heuristics, which are Cheapest-Insert Heuristic (CI), Nearest-Neighbor Heuristic (NN), Two-Directional Nearest-Neighbor Heuristic (2NN), Assignment-Patching Heuristic (AP), Nearest-Neighbor-Patching Heuristic (NNP), Two-Directional Nearest-Neighbor-Patching Heuristic (2NNP) and Greedy Heuristic (GR). Furthermore, we consider for each heuristic a version, where the heuristic is followed by the 3-OPT Heuristic.

A natural task regarding the quality of a heuristic for a given instance is to compare the value of the tour computed by the heuristic with the optimal value. For this reason, in our experiments we only used instances with a smaller number of nodes so that in most cases one of the exact algorithms is able to compute the optimal value. In particular we chose the number of nodes 10, 20, 30, 40, 50 for the random classes and instances with $n \in \{10, 20, \dots, 100\}$ for the real-world classes. For the asymmetric and symmetric classes 1 and 2 we only computed the exact values for $n \le 30$. Because we were not able to compute the optimal values for $n > 30$, we only compare the average values of the solutions found by the heuristics. For the symmetric classes 3 and 4 as well as for the instances from bioinformatics we were able to determine all optimal solutions by the BnC-E algorithm. For these instances we present the gaps to the optimal value computed by $gap = (heu - opt)/opt$, where heu and opt denote the heuristic value and the optimal value, respectively. The results can be found in Tables 1–4. The best values are emphasized.

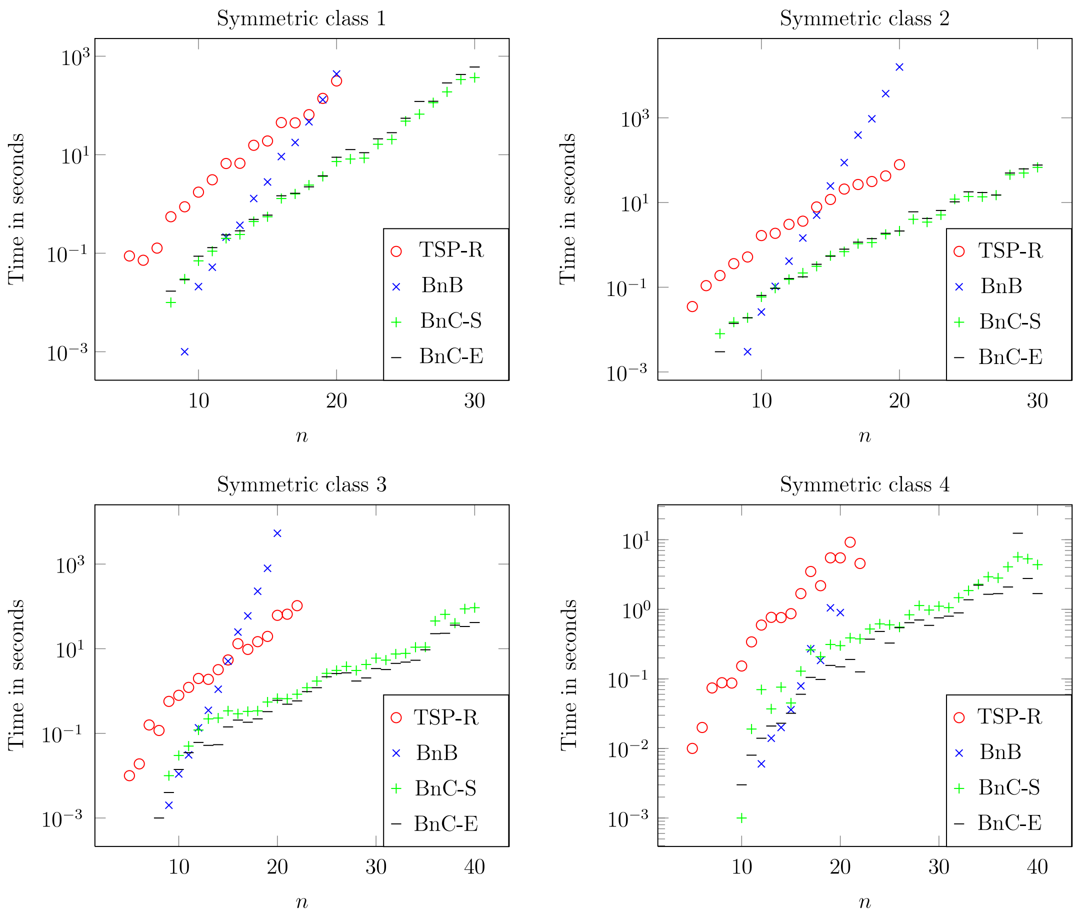


Fig. 3. Time for the exact algorithms applied to the symmetric random classes.

Table 1
Tour lengths for the heuristics applied to the asymmetric classes.

Size	Asymmetric class 1					Asymmetric class 2				
	10	20	30	40	50	10	20	30	40	50
CI	**21 496**	32 371	41 397	48 974	56 148	**263**	**2471**	**9 354**	**24 988**	**53 265**
NN	26 340	32 615	37 712	40 470	43 648	544	5779	23 002	62 157	125 408
2NN	24 422	31 349	36 030	39 361	41 502	518	6044	23 429	65 976	139 659
AP	34 208	71 121	112 836	151 693	196 960	453	6045	28 383	93 502	236 753
NNP	22 452	29 211	35 474	41 509	45 511	348	3555	14 362	38 676	79 407
2NNP	23 383	**29 167**	**32 905**	**37 395**	**40 125**	423	3908	14 675	41 325	80 450
GR	26 695	44 379	60 912	73 472	84 602	561	7273	34 391	104 371	231 228
CI + OPT	17 953	25 255	31 619	35 986	41 829	**179**	1584	5 594	14 417	29 974
NN + OPT	17 767	24 175	28 257	30 991	33 516	189	1491	5 389	13 980	27 944
2NN + OPT	**17 367**	23 595	**27 224**	**29 974**	**32 775**	184	1549	5 536	13 845	29 191
AP + OPT	18 280	25 487	32 720	37 775	44 141	182	1523	5 693	14 403	30 486
NNP + OPT	17 715	23 705	32 475	32 475	36 340	180	1515	5 514	**13 703**	28 750
2NNP + OPT	17 589	**23 370**	27 489	30 981	33 189	184	1508	**5 356**	13 780	**27 860**
GR + OPT	17 503	25 552	30 559	37 198	40 839	182	**1479**	5 563	14 056	28 863
Exact	12 403	11 606	10 998			120	705	2 144		

Running times of the heuristics:

The experiments show that all basic heuristics are rather fast for both, the random and the real-world instances, which is reasonable, as they have complexity not worse than $\mathcal{O}(n^3)$. Even for the largest instances the heuristics took not more than 10 s. For this reason, we do not present the exact running times here. Naturally, the heuristics with OPT steps are a bit slower.

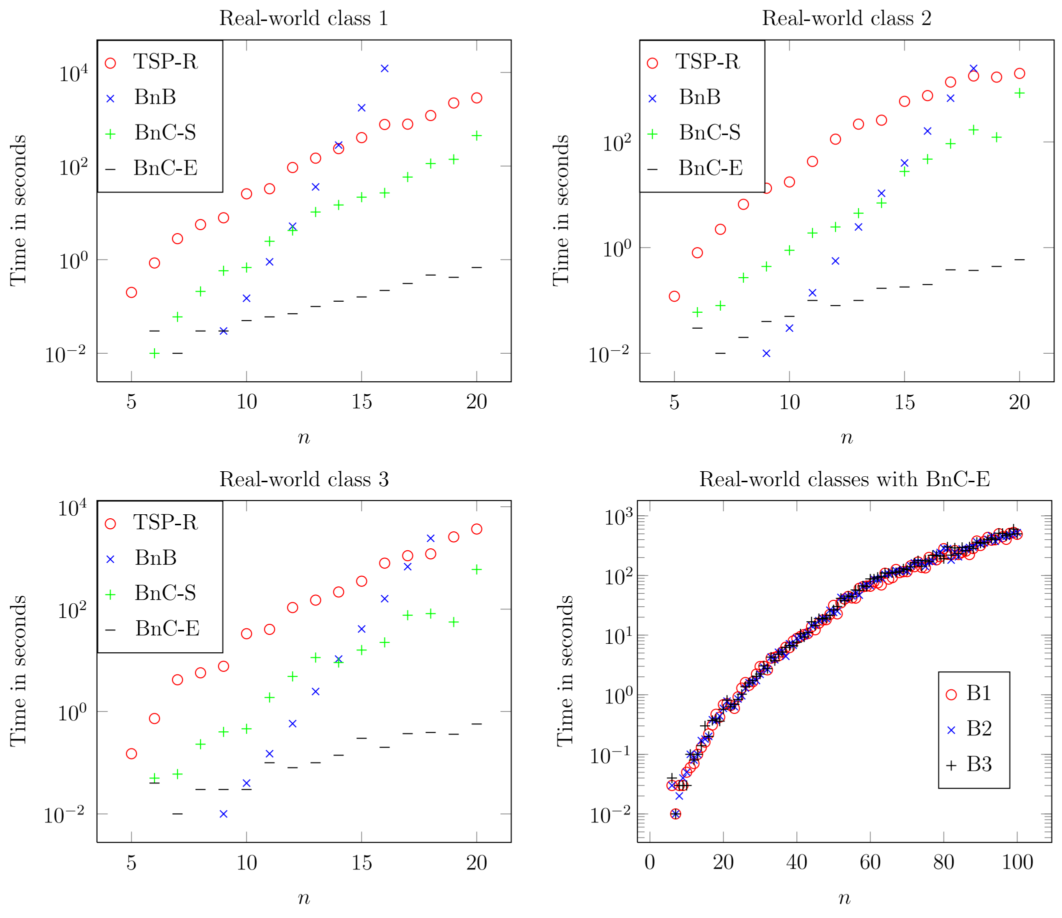


Fig. 4. Time for the exact algorithms applied to the real-world classes.

Table 2
Tour lengths for the heuristics applied to the symmetric classes 1 and 2.

Size	Symmetric class 1					Symmetric class 2				
	10	20	30	40	50	10	20	30	40	50
CI	**22 231**	32 730	41 990	48 407	55 956	**264**	**2453**	**9 526**	**24 775**	**53 023**
NN	25 370	33 246	37 677	41 775	42 630	551	5646	22 316	60 593	125 539
2NN	23 698	30 793	35 535	39 328	**40 694**	542	6400	24 035	62 148	134 717
AP	34 247	73 756	112 530	154 579	196 941	490	5632	30 877	97 347	239 418
NNP	22 829	30 195	37 487	41 579	45 831	390	3651	14 852	39 812	83 239
2NNP	23 174	**28 583**	**33 744**	**37 890**	41 229	460	4015	14 649	36 839	77 477
GR	27 706	45 251	60 205	73 056	85 281	542	7345	35 107	103 627	234 193
CI + OPT	16 809	21 356	26 374	30 702	34 593	167	1341	4 737	11 818	24 394
NN + OPT	16 699	21 594	24 433	26 885	28 617	165	**1282**	4 609	11 242	**22 819**
2NN + OPT	**16 520**	20 990	23 959	**26 777**	**28 051**	167	1343	4 517	11 348	22 909
AP + OPT	16 691	21 613	26 625	30 679	34 948	164	1304	4 646	11 829	24 190
NNP + OPT	16 851	20 928	24 702	28 179	30 273	**162**	1285	4 612	11 385	24 018
2NNP + OPT	16 755	20 820	**23 745**	27 168	29 826	167	1345	4 450	**10 999**	23 349
GR + OPT	16 567	**20 564**	25 605	29 513	32 900	167	**1282**	**4 438**	11 178	23 124
Exact	13 840	12 380	11 578			137	784	2 249		

Quality of the tours generated by the heuristics:

As expected, the OPT versions clearly beat the basic versions. Furthermore, the results for the (asymmetric and symmetric) random classes 1, the (asymmetric and symmetric) random classes 2, the symmetric random classes 3 and 4 and the real-world classes are completely different. For the random classes 1, 2NNP is the best heuristic, for the random classes 2–4, CI is the best heuristic, whereas for the real-world classes in average NN and 2NN are the best heuristics. AP is the worst heuristic for the random instances of class 1, and GR is the worst heuristic for the remaining classes. Considering the OPT versions we observe the following. For the random classes 1, 2NN is the best heuristic, whereas CI is the best heuristic for

Table 3
Average gap to optimal solution for the heuristics applied to the symmetric classes 3 and 4.

Size	Symmetric class 3					Symmetric class 4				
	10	20	30	40	50	10	20	30	40	50
CI	**0.0570**	**0.2049**	**0.3045**	**0.3776**	**0.4205**	0.0544	0.0950	**0.0956**	**0.1151**	**0.1090**
NN	0.2135	0.3296	0.4110	0.4684	0.5121	0.0798	0.1588	0.1880	0.1969	0.2049
2NN	0.2227	0.3763	0.4543	0.5010	0.5446	0.0584	0.1326	0.1765	0.1884	0.1946
AP	0.2363	0.5494	0.8828	1.2001	1.5970	**0.0426**	**0.0768**	0.1074	0.1283	0.1474
NNP	0.1780	0.3531	0.4649	0.5729	0.6482	0.0926	0.1579	0.2099	0.2568	0.2867
2NNP	0.1772	0.3275	0.4296	0.5079	0.5578	0.0635	0.1464	0.2034	0.2318	0.2479
GR	0.2332	0.5078	0.6736	0.8339	0.9731	0.1165	0.2282	0.2713	0.2908	0.3258
CI + OPT	**0.0039**	**0.0495**	**0.0607**	**0.0857**	**0.0985**	0.0007	0.0190	0.0234	0.0277	0.0306
NN + OPT	0.0136	0.0855	0.1311	0.1656	0.2019	**0.0005**	0.0104	0.0196	**0.0217**	0.0276
2NN + OPT	0.0189	0.0824	0.1354	0.1796	0.2078	0.0008	0.0138	0.0199	0.0250	0.0265
AP + OPT	0.0166	0.0753	0.1006	0.1482	0.1881	0.0009	0.0124	0.0165	0.0262	**0.0218**
NNP + OPT	0.0158	0.0690	0.0940	0.1306	0.1649	0.0010	**0.0068**	0.0195	0.0241	0.0242
2NNP + OPT	0.0188	0.0798	0.1150	0.1535	0.1781	0.0012	0.0132	0.0194	0.0247	0.0261
GR + OPT	0.0153	0.0758	0.0978	0.1456	0.1890	0.0009	0.0115	**0.0147**	0.0236	0.0253

Table 4
Average gap to optimal solution for the heuristics applied to the real-world classes.

Size	Real-world classes									
	10	20	30	40	50	60	70	80	90	100
CI	**0.0003**	**0.0216**	**0.0240**	0.0333	0.0314	0.0234	0.0228	0.0212	0.0208	0.0267
NN	0.0961	0.0395	0.0304	0.0305	0.0299	**0.0204**	**0.0170**	**0.0149**	**0.0142**	**0.0138**
2NN	0.0961	0.0395	0.0304	0.0305	0.0806	**0.0204**	**0.0170**	**0.0149**	**0.0142**	**0.0138**
AP	0.0145	0.0373	0.0265	**0.0235**	**0.0232**	0.0345	0.0289	0.0186	0.0433	0.0545
NNP	0.1141	0.0908	0.1156	0.1238	0.1078	0.1224	0.1339	0.1685	0.1515	0.1688
2NNP	0.0862	0.1026	0.1004	0.1163	0.1656	0.1693	0.1955	0.1847	0.2009	0.1926
GR	0.2548	0.2775	0.1705	0.2056	0.2855	0.2568	0.2775	0.2897	0.3512	0.3304
CI + OPT	0.0003	0.0003	0.0196	0.0002	0.0002	**0.0003**	**0.0003**	0.0003	**0.0003**	**0.0003**
NN + OPT	0.0003	0.0003	0.0008	**0.0001**	**0.0001**	**0.0003**	**0.0003**	0.0003	**0.0003**	**0.0003**
2NN + OPT	0.0003	0.0003	**0.0002**	**0.0001**	0.0554	**0.0003**	**0.0003**	0.0003	**0.0003**	**0.0003**
AP + OPT	**0.0000**	0.0003	0.0110	0.0195	0.0086	0.0148	0.0128	**0.0002**	0.0118	0.0155
NNP + OPT	**0.0000**	**0.0000**	0.0407	0.0266	0.0357	0.0280	0.0239	0.0342	0.0204	0.0620
2NNP + OPT	0.0243	0.0392	0.0149	0.0170	0.0719	0.0507	0.0498	0.0015	0.0372	0.0389
GR + OPT	0.0361	0.0185	0.0261	0.0173	0.0301	0.0328	0.0444	0.0406	0.0293	0.0284

the symmetric class 3, NN, 2NN are the best heuristics for the real-world classes, and the results are mixed for the random classes 2 and 4. Surprisingly, for several instances of the symmetric class 2, where GR is the worst heuristic for the basic versions, it is the best one regarding the OPT versions. Conversely, for several instances of the symmetric class 2, where CI is the best heuristic for the basic versions, it is the worst one regarding the OPT versions. This could mean that sometimes the OPT versions benefit from worse starting tours or, even stronger, that it is counterproductive to have good starting tours.

Regarding the comparison with the exact values, we have to realize that the heuristics presented in this paper do not behave very well applied to random instances of the classes 1 and 2. They generate tours whose costs are considerably larger than the costs of optimal tours. Considering the symmetric classes 3 and 4 and the real-world instances, things look nicer. For all instances we receive upper bounds very close to the optimum.

8. Summary and future research

In this paper we introduce the so called quadratic traveling salesman problem and present several exact and heuristical solution approaches. QTSP is a generalization of the TSP and has important applications in bioinformatics. Furthermore, special cases such as the angular-metric TSP or the TSP with reload cost have applications in robotics as well as in the planning of telecommunication and transport networks.

Regarding the tested real-world instances from bioinformatics, the exact algorithm based on branch-and-cut improved by certain cutting planes works surprisingly well. This indicates that these instances have some internal structure that should be studied in future work. Without the new cutting planes or using a branch-and-bound algorithm the running times are higher by several orders of magnitude.

From a practical perspective, there is demand for using P(VL)M models of higher order. In fact, PVLM models have been proposed as generalizations of PM models with the goal of modeling higher-order dependencies. Increasing the order of the P(VL)M models above 2 leads to traveling salesman problems with cubic or higher degree polynomial objective functions. If the degree of the polynomials increases, the linearization of all terms in the objective function leads to a very large number of variables. Here, better approaches leading to tractable model sizes must be developed, which is one of the objectives of *Polynomial Optimization*.

It has been shown in [23] that for each n there are infinitely many instances for which the NN (Nearest-Neighbor) and GR (Greedy) heuristic, respectively, find the unique worst possible tour, i.e., have domination number 1. In [5] a characterization of such cases for greedy type heuristics has been given. Thus it would be interesting to investigate whether this holds also for the NN and GR heuristic, respectively, for the QTSP.

Acknowledgments

We thank IPK Gatersleben, JKI Quedlinburg, and UZI Halle-Wittenberg for financial support.

Appendix

Proof of Theorem 4.1. We show the $\mathcal{NP}$-completeness of QCCP by a reduction from SAT [14].

Let an arbitrary SAT instance be given with variables x_i, $i \in N := \{1, \ldots, \bar{n}\}$, and clauses C_j, $j \in M := \{1, \ldots, m\}$, where the clauses consist of literals x_i and $\neg x_i$. Now we construct an instance of the asymmetric quadratic cycle cover problem. Let $G = (V, A)$ be a directed graph with set of nodes V and set of arcs A to be defined next. Instead of working with a complete graph we only specify those arcs and the weights of the 2-arcs, respectively, that are needed for the construction. The set V consists of the nodes:

1. $v_{i,j} \in V$ for all $i \in N, j \in \{0, \ldots, m\}$,
2. $u_j \in V$ for all $j \in M$.

Thus $m + 1$ nodes exist for each variable, and there is one additional node for each clause. The set of arcs of G can be partitioned into several subsets. For each variable x_i, $i \in N$, the graph contains the sets of arcs

- A_i being built of
  - $(v_{i,m}, v_{i,0})$,
  - $(v_{i,j-1}, v_{i,j})$ for all $j \in M$,
  - $(v_{i,j-1}, u_j)$ and $(u_j, v_{i,j})$ if the literal x_i appears in the clause C_j,
- and $A_i^{\neg}$ containing
  - $(v_{i,0}, v_{i,m})$,
  - $(v_{i,j}, v_{i,j-1})$ for all $j \in M$,
  - $(v_{i,j}, u_j)$ and $(u_j, v_{i,j-1})$ if the literal $\neg x_i$ appears in the clause C_j.

The set of all arcs is $A = \bigcup_{i \in N}(A_i \cup A_i^{\neg})$. This leads to $A^3 := \{ijk \in V^{(3)} : ij, jk \in A^{(2)}\}$.

With this definition of A^3 it holds

$$K_1^i = \{v_{i,0}v_{i,1}v_{i,2}, v_{i,1}v_{i,2}v_{i,3}, \ldots, v_{i,m-1}v_{i,m}v_{i,0}, v_{i,m}v_{i,0}v_{i,1}\} \subset A^3$$

as well as

$$K_2^i = \{v_{i,m}v_{i,m-1}v_{i,m-2}, v_{i,m-1}v_{i,m-2}v_{i,m-3}, \ldots, v_{i,1}v_{i,0}v_{i,m}, v_{i,0}v_{i,m}v_{i,m-1}\} \subset A^3$$

for each variable x_i, $i \in N$ (here cycles are represented by their 2-arcs). Furthermore, if the literal x_i, $i \in N$, appears in the clauses C_J, $J \subset \{j \in M : x_i \in C_j\}$, we can enlarge cycle K_1^i to $\bar{K}_1^{i,J}$ by replacing for each $j \in J$ the path $(v_{i,j-1}, v_{i,j})$ with the path $(v_{i,j-1}, u_j, v_{i,j})$, *e.g.*,

$$\bar{K}_1^{i,\{j\}} = \{v_{i,0}v_{i,1}v_{i,2}, \ldots, v_{i,j-3}v_{i,j-2}v_{j-1}, v_{i,j-2}v_{i,j-1}u_j, v_{i,j-1}u_jv_{i,j}, u_jv_{i,j}v_{i,j+1}, v_{i,j}v_{i,j+1}v_{i,j+2}, \ldots, v_{i,m}v_{i,0}v_{i,1}\} \subset A^3$$

Similarly, if the literal $\neg x_i$, $i \in N$, appears in the clauses $C_{\bar{J}}, \bar{J} \subset \{j \in M : \neg x_i \in C_j\}$, we can enlarge cycle K_2^i to $\bar{K}_2^{i,\bar{J}}$ by replacing for each $j \in \bar{J}$ the path $(v_{i,j}, v_{i,j-1})$ with the path $(v_{i,j}, u_j, v_{i,j-1})$, *e.g.*,

$$\bar{K}_2^{i,\{j\}} = \{v_{i,m}v_{i,m-1}v_{i,m-2}, \ldots, v_{i,j+2}v_{i,j+1}v_{i,j}, v_{i,j+1}v_{i,j}u_j,$$
$$v_{i,j}u_jv_{i,j-1}, u_jv_{i,j-1}v_{i,j-2}, v_{i,j-1}v_{i,j-2}v_{i,j-3}, \ldots, v_{i,0}v_{i,m}v_{i,m-1}\} \subset A^3$$

As an example, Fig. 5 shows the graph for the SAT instance

$$(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2).$$

Obviously, the construction above is possible in polynomial time.

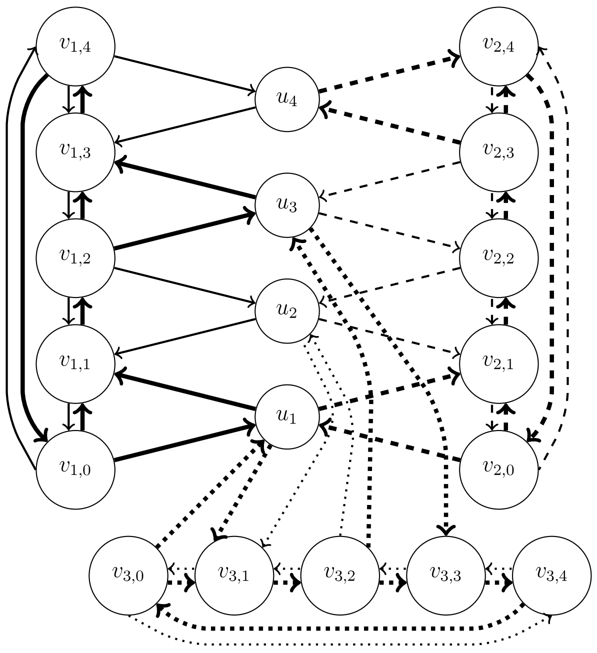


Fig. 5. Visualization of the construction used in the proof of Theorem 4.1 for the SAT-formula $(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2)$. All arcs that correspond to *true* are drawn with bold lines. The arcs with solid lines correspond to x_1, $\neg x_1$, the arcs with dashed line to x_2, $\neg x_2$ and the dotted lines to x_3, $\neg x_3$.

The goal of QCCP is to cover all nodes with cycles with minimal total costs. If the arcs of A_i, $i \in N$, (the arcs with thicker lines in Fig. 5) are used, this corresponds to setting the variable x_i to *true*, and if the arcs of $A_i^{\neg}$ are used, this corresponds to setting the variable x_i to *false*. We have to ensure that in an optimal solution of QCCP only arcs of exactly one of those two sets are used for each variable x_i and that there are no paths consisting of arcs of different variables (if this is possible at all). Therefore, we use the following cost function for the associated 2-arcs. We set $c_q(p, q, r) = 0$, $(p, q, r) \in V^{(3)}$, if and only if $pq, qr \in A_i$ or $pq, qr \in A_i^{\neg}$ for one $i \in N$ and $c_q(p, q, r) = 1$ otherwise. With this construction it is also possible to work with a complete directed graph setting all weights of 2-arcs to one except for paths in A_i or $A_i^{\neg}$, $i \in N$.

It remains to show that all nodes may be covered by cycles such that for each $i \in N$ only arcs either from A_i or from $A_i^{\neg}$ are used, but never from both (which yields an objective value of 0), if and only if the SAT instance is satisfiable.

- Let $f : N \to \{true, false\}$ be a satisfying assignment of the variables (*i.e.*, setting $x_i = f(i)$ satisfies all clauses). Since f satisfies all clauses, for each $C_j, j \in M$, there is (at least) one literal $x_{i(j)}$ or $\neg x_{i(j)}$ contained in C_j that is set to true ($i(j)$ denotes the index of one of those variables making C_j true). We construct one cycle with costs of 0 for each variable. Let x_i, $i \in N$, be an arbitrary variable.
  - If $f(i) = true$,
    - if $J = \{j \in M : i(j) = i\} \neq \emptyset$, use $\bar{K}_1^{i,J}$ with $(\bar{K}_1^{i,J})^{(2)} \subset A_i$ (the mentioned cycle-enlargement),
    - otherwise use K_1^i with $(K_1^i)^{(2)} \subset A_i$.
  - If $f(i) = false$,
    - if $\bar{J} = \{j \in M : i(j) = i\} \neq \emptyset$, use $\bar{K}_2^{i,\bar{J}}$ with $(\bar{K}_2^{i,\bar{J}})^{(2)} \subset A_i^{\neg}$,
    - otherwise use K_2^i with $(K_2^i)^{(2)} \subset A_i^{\neg}$.

  Then each node lies exactly on one of these $\bar{n}$ cycles and the total weight of these cycles is zero.
- On the other hand, let $W_1, \ldots, W_p$ be the cycles of a QCCP-solution with total costs of zero. By construction of the graph and the weights, the arcs of each cycle belong to exactly one set A_i or $A_i^{\neg}$. So, w.l.o.g., we may assume $p = \bar{n}$ and $W_i^{(2)} \subset A_i$ or $W_i^{(2)} \subset A_i^{\neg}$. We set x_i to *true* if $W_i^{(2)} \subset A_i$ and we set x_i to *false* if $W_i^{(2)} \subset A_i^{\neg}$. Since the cycles cover all nodes, for each $u_j, j \in M$, there is a cycle $W_{i(j)}$ such that $u_j \in V(W_{i(j)})$. Let $C_j, j \in M$, be an arbitrary clause. If $W_{i(j)}^{(2)} \subset A_{i(j)}$, the literal $x_{i(j)}$ has been set to *true* and appears in clause C_j by the construction of the graph. Analogously, if $W_{i(j)}^{(2)} \subset A_{i(j)}^{\neg}$, the literal $\neg x_{i(j)}$ has been set to *true* and appears in clause C_j. Consequently, the truth assignment constructed above fulfills all clauses. □

Fig. 6 shows a QCCP solution for the SAT instance

$$(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2)$$

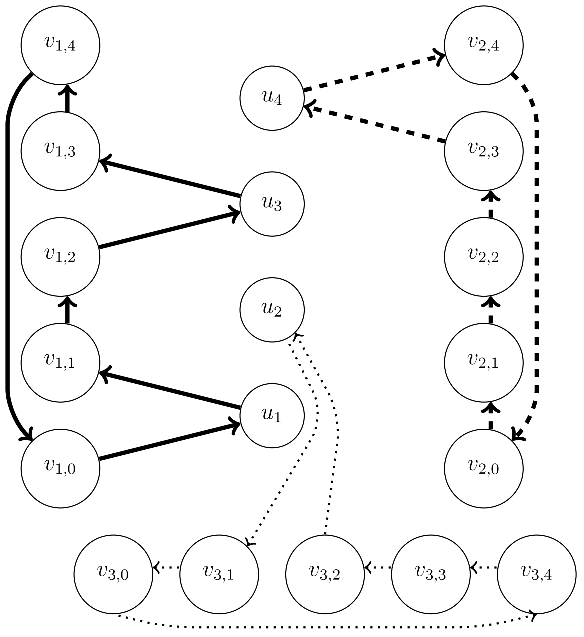


Fig. 6. A possible cycle cover solution for the instance $(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2)$ for the truth assignment $x_1 = \mathit{true}, x_2 = \mathit{true}, x_3 = \mathit{false}$.

with truth assignment. The solid lines correspond to $x_1 = \mathit{true}$, the dashed lines to $x_2 = \mathit{true}$ and the dotted lines to $x_3 = \mathit{false}$. The complete construction used in the proof of Theorem 4.1 can be seen in Fig. 5.

References

[1] A. Aggarwal, D. Coppersmith, S. Khanna, R. Motwani, B. Schieber, The angular-metric traveling salesman problem, SIAM Journal on Computing 29 (1999) 697–711.

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

[3] D.L. Applegate, R.E. Bixby, V. Chvatal, W.J. Cook, The Traveling Salesman Problem: A Computational Study, in: Princeton Series in Applied Mathematics, Princeton University Press, 2007.

[4] D.L. Applegate, R.E. Bixby, V. Chvátal, W. Cook, D.G. Espinoza, M. Goycoolea, K. Helsgaun, Certification of an optimal TSP tour through 85,900 cities, Operations Research Letters 37 (1) (2009) 11–15.

[5] J. Bang-Jansen, G. Gutin, A. Yeo, When the greedy algorithm fails, Discrete Optimization 1 (2004) 121–127.

[6] A. Behzad, M. Modarres, New efficient transformation of the generalized traveling salesman problem into traveling salesman problem, in: Proceedings of the 15th International Conference of Systems Engineering, Las Vegas, 2002.

[7] Concorde TSP Solver. Information available on http://www.tsp.gatech.edu/concorde.html.

[8] G. Dantzig, R. Fulkerson, S. Johnson, Solution of a large-scale traveling-salesman problem, Operations Research 2 (1954) 393–410.

[9] V. Dimitrijević, Z. Šarić, An efficient transformation of the generalized traveling salesman problem into the traveling salesman problem on digraphs, Information Sciences 102 (1–4) (1997) 105–110.

[10] K. Ellrott, C. Yang, F.M. Sladek, T. Jiang, Identifying transcription factor binding sites through Markov chain optimization, Bioinformatics 18 (Suppl 2) (2002) 100–109.

[11] A. Fischer, An analysis of the asymmetric quadratic traveling salesman polytope, SIAM Journal on Discrete Mathematics (2013) (in press).

[12] M. Fischetti, J.-J. Salazar-González, P. Toth, The generalized traveling salesman and orienteering problems, in: G. Gutin, A.P. Punnen (Eds.), The Traveling Salesman Problem and its Variations, in: Combinatorial Optimization, Kluwer Academic Publishers, 2002, pp. 609–662.

[13] A.M. Frieze, J. Yadegar, On the quadratic assignment problem, Discrete Applied Mathematics 5 (1) (1983) 89–98.

[14] M.R. Garey, D.S. Johnson, Computers and Intractability, A Guide to the Theory of NP-Completeness, W.H. Freeman and Company, New York, 1979.

[15] F. Glover, G. Gutin, A. Yeo, A. Zverovich, Construction heuristics for the asymmetric TSP, European Journal of Operational Research 129 (2000) 555–568.

[16] B. Goldengorin, G. Jäger, G. Gutin, D. Ghosh, Tolerance-based Algorithms for the Traveling Salesman Problem, World Scientific, 2008, pp. 47–59 (Chapter 5).

[17] B. Goldengorin, G. Jäger, P. Molitor, Some basics on tolerances, in: S.-W. Cheng, C.K. Poon (Eds.), AAIM, in: Lecture Notes in Computer Science, vol. 4041, Springer, 2006, pp. 194–206.

[18] B. Goldengorin, G. Jäger, P. Molitor, Tolerance based contract-or-patch heuristic for the asymmetric TSP, in: T. Erlebach (Ed.), CAAN, in: Lecture Notes in Computer Science, vol. 4235, Springer, 2006, pp. 86–97.

[19] M. Grötschel, O. Holland, Solution of large-scale symmetric travelling salesman problems, Mathematical Programming. Series A 51 (2) (1991) 141–202.

[20] M. Grötschel, L. Lovász, A. Schrijver, Geometric Algorithms and Combinatorial Optimization, second corrected ed., in: Algorithms and Combinatorics, vol. 2, Springer, 1993.

[21] G. Gutin, A. Punnen (Eds.), The Traveling Salesman Problem and its Variations, in: Combinatorial Optimization, vol. 12, Kluwer Academic Publishers, 2002.

[22] G. Gutin, A. Yeo, Assignment problem based algorithms are impractical for the generalized TSP, Australasian Journal of Combinatorics 27 (2003) 149–154.

[23] G. Gutin, A. Yeo, A. Zverovich, Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the TSP, Discrete Applied Mathematics 117 (2002) 81–86.

[24] K. Helsgaun, An effective implementation of the Lin-Kernighan traveling salesman heuristic, European Journal of Operational Research 126 (2000) 106–130.

[25] S. Hong, A linear programming approach for the traveling salesman problem, Ph.D. Thesis, John Hopkins University, Baltimore, Maryland, USA, 1972.
[26] IBM ILOG CPLEX 12.1: Using the CPLEX callable library. Information available on http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/.
[27] R. Jonker, T. Volgenant, Transforming asymmetric into symmetric traveling salesman problems, Operations Research Letters 2 (4) (1983) 161–163.
[28] R. Jonker, A. Volgenant, A shortest augmenting path algorithm for dense and sparse linear assignment problems, Computing 38 (4) (1987) 325–340.
[29] R. Karp, J. Steele, Probabilistic analysis of heuristics, in: E.L. Lawler, J.K. Lenstra, A.H.G.R. Kan, D.B. Shmoys (Eds.), The Traveling Salesman Problem. A Guided Tour of Combinatorial Optimization, John Wiley & Sons, Chichester, 1985, pp. 181–206 (Chapter 6).
[30] LEMON Graph Library 1.2.2. Information available at http://lemon.cs.elte.hu/trac/lemon.
[31] S. Lin, B.W. Kernighan, An effective heuristic algorithm for the traveling-salesman problem, Operations Research 21 (2) (1973) 498–516.
[32] D.G. Macharet, A.A. Neto, V.F. da Camara Neto, M.F.M. Campos, Nonholonomic path planning optimization for Dubins vehicles, in: IEEE International Conference on Robotics and Automation, ICRA 2011, Shanghai, China, 9–13 May 2011, IEEE, 2011, pp. 4208–4213.
[33] A.C. Medeiros, S. Urrutia, Discrete optimization methods to determine trajectories for Dubins' vehicles, Electronic Notes in Discrete Mathematics 36 (2010) 17–24. ISCO 2010—International Symposium on Combinatorial Optimization.
[34] G.L. Nemhauser, L.A. Wolsey, Integer and Combinatorial Optimization, Wiley-Interscience, New York, NY, USA, 1988.
[35] C.E. Noon, J.C. Bean, An efficient transformation of the generalized traveling salesman problem, Technical Report, University of Michigan, 1991.
[36] M. Padberg, G. Rinaldi, A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems, SIAM Review 33 (1) (1991) 60–100.
[37] D.J. Rosenkrantz, R.E. Stearns, P.M. Lewis II, An analysis of several heuristics for the traveling salesman problem, SIAM Journal on Computing 6 (3) (1977) 563–581.
[38] K. Savla, E. Frazzoli, F. Bullo, Traveling salesperson problems for the Dubins vehicle, IEEE Transactions on Automatic Control 53 (6) (2008) 1378–1391.
[39] A. Schrijver, Theory of Linear and Integer Programming, Wiley, 2000.
[40] X. Zhao, H. Huang, T.P. Speed, Finding short DNA motifs using permuted Markov models, Journal of Computational Biology 12 (6) (2005) 894–906.