Lower bounding procedure for the Asymmetric Quadratic Traveling Salesman Problem

Borzou Rostami[a,*], Federico Malucelli[b], Pietro Belotti[c], Stefano Gualandi[d]

[a] *Fakultät für Mathematik, TU Dortmund, Germany*
[b] *Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milan, Italy*
[c] *Xpress Development Team, FICO, Birmingham, UK*
[d] *AntOptima, SA, Lugano, Switzerland*

Abstract

In this paper we consider the Asymmetric Quadratic Traveling Salesman Problem. Given a directed graph and a function that maps every pair of consecutive arcs to a cost, the problem consists in finding a cycle that visits every vertex exactly once and such that the sum of the costs is minimum. We propose an extended Linear Programming formulation that has a variable for each cycle in the graph. Since the number of cycles is exponential in the graph size, we propose a column generation approach. We compare the bounds resulting from this new formulation with those obtained by some linearization techniques for 0-1 quadratic optimization or specifically proposed for the QTSP. Computational results on some set of benchmarks used in the literature show that the column generation approach is very promising.

Keywords: Traveling salesman, Integer programming, Quadratic Traveling Salesman Problem, Column Generation, Lower bound, Cycle cover.

1. Introduction

The Traveling Salesman Problem (TSP) is one of the most studied optimization problems. Given a graph $G = (V, A)$ with costs $c_e, e \in A$, the problem consists in finding a cycle C that visits each vertex in V exactly once (i.e., a Hamiltonian cycle), and such that the total cycle cost is minimum. In its most common form, the TSP has a linear cost function, equal to the sum of the costs c_e of each edge in C. Now, consider a variant of the TSP having a non-negative interaction cost Q_{ef} for every pair of consecutive arcs $e, f \in A$. Because the cost of a cycle can be computed as a quadratic function, this problem is called Quadratic TSP (QTSP) and is NP-hard as shown in [1]. As for the linear counterpart, the QTSP has a symmetric and an asymmetric variants (SQTSP and AQTSP, respectively). In this paper we focus on the asymmetric version which appears to be more challenging.

*Email address: brostami@mathematik.tu-dortmund.de

The QTSP was introduced with an application to bioinformatics in [2] and also has application in robotics and telecommunications. In robotics, a variant of QTSP called *Angle-TSP* can be used for the optimization of robot paths with respect to energetic aspects. The Angle-TSP problem seeks to minimize the total angle of a TSP tour for a set of points in the Euclidean space where the angle of a tour is the sum of the direction changes at all points [3]. Also the QTSP can be viewed as a generalization of the Reload Cost TSP (RTSP) introduced in [4]. In the RTSP, one is given a graph whose every edge is assigned a color and there is a *reload* cost when passing through a node on two edges that have different colors.

The QTSP has been introduced quite recently and its literature is rather limited. In particular, the QTSP has been tackled in [1] with heuristic algorithms based on a well-known heuristics for the TSP, with an ad-hoc branch-and-bound, and with a branch-and-cut approach based on a linearization of a 0-1 Quadratic Programming formulation of the problem. In [5] and [6] the polyhedral structure of a linearized integer programming formulation has been used to develop a branch-and-cut algorithm for the SQTSP and AQTSP respectively that are the current state-of-the-art for QTSP.

In this paper we study a lower bounding procedure for the Asymmetric QTSP (AQTSP). Considering the special structure of the quadratic costs of the AQTSP, we present an extended Linear Programming formulation of the problem with an exponential number of variables that is solved via Column Generation (CG). The basic idea is to have a variable for each cycle of G. This yields a pricing subproblem that consists in finding a cycle of minimum quadratic cost. We formulate the pricing subproblem as a 0-1 Quadratic Program, which is linearized and solved with standard techniques. We resort to stabilization techniques to overcome the tailing-off effect of the CG approach. We also present the linearized integer programming formulation proposed in [6] and show how it can be obtained using a partial application of the Reformulation-Linearization technique (RLT) [7, 8]. This formulation is, in fact, based on the standard Asymmetric Traveling Salesman Problem (ATSP) formulation of Dantzig et al. [9] which has an exponential number of constraints. In order to avoid this complication we use the compact formulation of Miller, Tucker and Zemlin [10] with a polynomial number of variables and constraints. In order to generate superior bounds, we also develop a mixed 0-1 linear formulation based on using the first-level RLT and show how to handle it via a Lagrangian relaxation where the Lagrangian function has block-diagonal structure. Our computational experiments show that the column generation approach outperforms all other lower bounding schemes.

2. Problem formulation

Consider a complete directed graph $G = (V, A)$ with vertex set $V = \{1, 2, \ldots, n\}$, arc set A, and cost function Q that maps every pair of consecutive arcs to a non negative integer cost. The AQTSP seeks a directed tour (i.e., a directed cycle passing through each vertex exactly once) of minimum cost. Let binary variable x_{ij} be equal to 1 if the arc $(i, j) \in A$

belongs to the minimum cost tour, and zero otherwise. The AQTSP then call for

$$\begin{aligned}
\text{AQTSP:} \quad \min \quad & \sum_{(i,j)\in A} \sum_{(j,k)\in A} Q_{ijk} x_{ij} x_{jk} & & \\
\text{s.t.} \quad & \sum_{j:(i,j)\in A} x_{ij} = 1 & \forall i \in V & \quad (1) \\
& \sum_{i:(i,j)\in A} x_{ij} = 1 & \forall j \in V & \quad (2) \\
& \sum_{\substack{i\in S, j\notin S: \\ (i,j)\in A}} x_{ij} \geq 1 & \forall S \subset V,\ 2 \leq |S| \leq n-2 & \quad (3) \\
& x_{ij} \in \{0,1\} & \forall (i,j) \in A. &
\end{aligned}$$

Constraints (1) and (2) force to select a single outgoing arc and a single incoming arc for each node, respectively, and constraints (3) are the well known subtour elimination constraints [9].

Note that the AQTSP formulation, presented here, is based on the ATSP formulation of Dantzig et al. [9] which has an exponential number of constraints. In order to avoid this complication we can take advantage of some known compact formulations of the ATSP, i.e., formulations with a polynomial number of both variables and constraints. Let us consider the MTZ formulation of Miller, Tucker and Zemlin [10] which fixes node 1 as a "depot", which the salesman must leave at the start of the tour and return to at the end of the tour. Let u_i be a continuous variable representing the position of node i in the tour. Then the MTZ formulation replace constraints (3) with the following constraints:

$$u_i - u_j + (n-1)x_{ij} \leqslant n-2 \qquad 2 \leqslant i, j \leqslant n-2;\ i \neq j \qquad (4)$$

$$1 \leqslant u_i \leqslant n-1 \qquad \forall i = 2, \ldots n, \qquad (5)$$

where constraints (4) ensure that, if the salesman travels from node i to node j, then the position of node j is one more than that of node i, and constraints (4) together with the bounds (5) ensure that each non-depot node is in a unique position.

3. Linearizations and lower bounds

The main difficulty of the AQTSP, in addition to the TSP problem structure, lies in the quadratic structure of the objective function. One of the most natural ways to obtain a lower bound is to linearize the quadratic terms $x_{ij}x_{jk}$ for all $(i,j),(j,k) \in A$ using the standard linearization technique of Glover and Woolsey [11]. The approach is based on introduction of new non-negative binary variables $y_{ijk} = x_{ij}x_{jk}$ which satisfy the following set of constraints:

$$y_{ijk} \leq x_{ij}, \quad y_{ijk} \leq x_{jk}, \quad \text{and} \quad y_{ijk} \geq x_{ij} + x_{jk} - 1.$$

The main disadvantage of the standard linearization is that it does not take the problem structure into account. In the following subsections we present some reformulations for the problem which take advantage of the TSP structure.

3.1. Cycle reformulation

In this section we present a new formulation of the AQTSP that considers an exponential number of variables, one for each cycle of the given graph. Let C be a cycle of G described as set of arcs. The cost of cycle C is the sum of the costs of the pairs of consecutive arcs contained in the cycle, i.e.

$$q(C) = \sum_{i,j,k \in V:(i,j),(j,k) \in C} Q_{ijk}.$$

Let $\mathcal{C}$ and $\mathcal{T}$ denote the collection of all cycles and all tours of G, respectively. Clearly, we have that $\mathcal{T} \subseteq \mathcal{C}$, and hence

$$\min_{C \in \mathcal{C}} q(C) \leq \min_{T \in \mathcal{T}} q(T).$$

Following the approach of Held and Karp to the TSP [12], we associate a penalty π_i with every vertex i in V, and define $\pi(C) = \sum_{i \in C} \pi_i$. Although C is a set of arcs, here we extend the notation so that $i \in C$ means $i \in V$ and $\exists j \in V : (i,j) \in C \vee (j,i) \in C$. Let us consider a new cost function defined as follows: $d(C) = q(C) + \pi(C)$. Let T^* denote an optimal tour of G, i.e. $q(T^*) = \min_{T \in \mathcal{T}} q(T)$. Then, the following relations hold:

$$\min_{C \in \mathcal{C}} d(C) = \min_{C \in \mathcal{C}} \{q(C) + \pi(C)\} \leq d(T^*) = q(T^*) + \pi(V)$$

$$\min_{C \in \mathcal{C}} \{q(C) - \pi(V \setminus C)\} \leq q(T^*).$$

For any vector of penalty terms π we get a lower bound. However, we are interested in finding π that maximizes the lower bound:

$$\max_{\pi \in \Re^n} \ \min_{C \in \mathcal{C}} \{q(C) - \pi(V \setminus C)\}, \tag{6}$$

which is linearized by introducing a variable z as follows:

$$\begin{aligned} \text{P: max} \quad & z & & \\ \text{s.t.} \quad & z + \pi(V \setminus C) \leq q(C) & \forall C \in \mathcal{C} & \quad (7) \\ & z, \pi \text{ unrestricted.} & & \end{aligned}$$

Let λ_C be the dual multipliers of constraints (7). We can write the dual of problem P as follows:

$$\begin{aligned} \text{D1: min} \quad & \sum_{C \in \mathcal{C}} q(C)\lambda_C & & \\ \text{s.t.} \quad & \sum_{C \in \mathcal{C}: i \notin C} \lambda_C = 0 & \forall i \in V & \quad (8) \\ & \sum_{C \in \mathcal{C}} \lambda_C = 1 & & \quad (9) \\ & \lambda_C \geq 0 & \forall C \in \mathcal{C}. & \quad (10) \end{aligned}$$

Since all multipliers in (8) are non-negative, an optimal solution must satisfy $\lambda_C^* = 1$ for some $C^* \in \mathcal{C}$ and $\lambda_C^* = 0$ for all $C \in \mathcal{C} \setminus C^*$. It follows that the cycle C^* (single or multiple) is optimal and $q(C^*)$ provides a lower bound for the original QTSP.

By subtracting each constraint (8) from (9) and removing (9) from D1, one can find a relaxation of the problem D1 as follows:

$$\begin{aligned} \text{D2: min} \quad & \sum_{C \in \mathcal{C}} q(C)\lambda_C & & \\ \text{s.t.} \quad & \sum_{C \in \mathcal{C}: i \in C} \lambda_C = 1 & \forall i \in V & \qquad (11) \\ & \lambda_C \geq 0 & \forall C \in \mathcal{C}. & \end{aligned}$$

Problem D2 seeks a minimum-weight "combination of cycles" such that each vertex appears, on average, in one cycle.

3.2. Reformulation-Linearization technique applied to the AQTSP

The Reformulation-Linearization technique (RLT), which was introduced in [7, 8] for general zero-one polynomial programs, transforms the original problem into a mixed 0-1 linear program via two basic steps of *reformulation* and *linearization*. In the reformulation step, the constraints are multiplied by the binary variables and their complements to construct redundant nonlinear constraints. In the linearization step, the objective and constraints of the reformulated problem are linearized by substituting a continuous variable for each distinct nonlinear term. Depending on how many times this process is applied, different levels of RLT can be obtained. In this section we concern ourselves with the application of the level-1 RLT for the AQTSP. For ease of notation we define the set $\mathcal{I} = \{(i,j,k,l) : (i,j) \in A, (k,l) \in A, i \neq k, j \neq l, (i,j) \neq (l,k)\}$. The level-1 RLT representation is generated via the following two steps:
Reformulation: Multiply each constraints (1)-(3) by each of the $|A|^2$ binary variables x_{kl}, and append these new constraints to the formulation. When the variable x_{ij} in a given constraint is multiplied by x_{kl}, express the resulting product as $x_{ij}x_{kl}$ in that order. Substitute x_{kl}^2 with x_{kl} throughout the constraints and set $x_{ij}x_{kl} = 0$ if $i = k$ and $j \neq l$ or $i \neq k$ and $j = l$. Moreover set $x_{ij}x_{ji} = 0$ to eliminate subtours involving only two nodes.
Linearization: For all $(i,j,k,l) \in A$, substitute each product $x_{ij}x_{kl}$ with y_{ijkl} and enforce the following equalities to the problem:

$$y_{ijkl} = y_{klij} \qquad \forall (i,j,k,l) \in \mathcal{I},\ i < k. \qquad (12)$$

The level-1 RLT results as follows:

$$
\begin{aligned}
\text{RLT1: } \min \quad & \sum_{(i,j,k,l)\in\mathcal{I}} D_{ijkl}y_{ijkl} & & \\
\text{s.t.} \quad & \sum_{j:(i,j,k,l)\in\mathcal{I}} y_{ijkl} = x_{kl} & & \forall i \in V,\ \forall (k,l)\in A,\ k\neq i \\
& \sum_{i:(i,j,k,l)\in\mathcal{I}} y_{ijkl} = x_{kl} & & \forall j \in V,\ \forall (k,l)\in A,\ k\neq j \\
& \sum_{i\in S, j\notin S:(i,j,k,l)\in\mathcal{I}} y_{ijkl} \geq x_{kl} & & S\subset V,\ 2\leq |S|\leq n-2,\ \forall (k,l)\in A \\
& y_{ijkl} = y_{klij} & & \forall (i,j,k,l)\in\mathcal{I},\ i<k \\
& y_{ijkl} \geq 0 & & (i,j,k,l)\in\mathcal{I} \\
& x\in\mathcal{X}, x\in\{0,1\} & &
\end{aligned}
$$

where

$$\mathcal{X} = \big\{0\leq x\leq 1:\ (1)-(3)\big\}$$

and $D_{ijkl} = Q_{ijl}$ for all $(i,j,k,l)\in\mathcal{I}$ with $i\neq l$ and $j=k$, Otherwise, $D_{ijkl}=0$.

Note that one can also multiply constraints (3) by $(1-x_{kl})$ as a complete application of RLT [7, 8]. However, to keep the block-diagonal structure of the problem which lends to efficient solution methods, we do not consider these inequalities.

In order to solve the LP relaxation of RLT1, we construct a Lagrangian dual by placing constraints (12) into the objective function as proposed by Adams and Johnson [13] for the Level-1 RLT representation of the QAP. To obtain an optimal solution $(\tilde{\boldsymbol{x}},\tilde{\boldsymbol{y}})$ to the resulting problem we can use the well known Gilmore-Lawler procedure [14, 15] as follows. We first solve $|A|$ LP relaxation of separate ATSP in G, one for each arc $(k,l)\in A$:

$$\bar{D}_{kl} = \min\left\{\sum_{(i,j)\in A:(i,j,k,l)\in\mathcal{I}} D_{ijkl}y_{ijkl}\ :\ \boldsymbol{y}_{kl}\in\mathcal{X}\right\}. \tag{13}$$

Then solving the following problem gives the lower bound:

$$\min\left\{\sum_{(k,l)\in A}\bar{D}_{kl}x_{kl}\ :\ x\in\mathcal{X}\right\}. \tag{14}$$

Once (14) is solved, the optimal solution of RLT1 is obtained as $\tilde{\boldsymbol{x}}=\bar{\boldsymbol{x}}$ and, $\tilde{\boldsymbol{y}}_{kl}=\tilde{x}_{kl}\bar{\boldsymbol{y}}_{kl}$ for $(k,l)\in A$. where $\bar{\boldsymbol{x}}$ and $\bar{\boldsymbol{y}}$ are the optimal solution of problems (14) and (13), respectively.

In our lower bound computation we used a dual-ascent strategy similar to those proposed in [16, 17] for solving the Lagrangian dual arising in RLT representations of the QAP. This approach uses the dual information retrieved from (13) and (14) to transform the problem to a series of equivalent AQTSP. Then it generates a monotonic nondecreasing sequence of

lower bounds on the LP relaxation of RLT1. Since each of problems (13) and (14) contains an exponential number of constraints (3), explicitly keeping track of the dual values for (3) is not practical. To overcome this problem, we consider a different relaxation of ATSP using the MTZ constraints (4) and (5). In fact we replace $\mathcal{X}$ with $\mathcal{X}'$ where

$$\mathcal{X}' = \big\{0 \le x \le 1 : \ (1), (2), (4), (5)\big\}.$$

Note that the only difference is that subtour elimination constraints (3) are replaced with constraints (4) and (5).

3.3. Fischer linearization and its compact form

To generate lower bounds, Fischer [6] introduced a mixed integer linear programming with $n(n-1)(n-2)/6$ continuous variables y_{ijk} such that

$$y_{ijk} = x_{ij}x_{jk} \text{ for all } (i,j),(j,k) \in A \text{ with } i \ne k.$$

The formulation has the following form:

$$\begin{aligned}
\text{LP1:} \quad \min \quad & \sum_{(i,j)\in A} \sum_{(j,k)\in A: k\ne i} Q_{ijk} y_{ijk} & & \\
\text{s.t.} \quad & \sum_{k:(j,k)\in A, k\ne i} y_{ijk} = x_{ij} & \forall (i,j) \in A & \quad (15)\\
& \sum_{k:(k,i)\in A: k\ne j} y_{kij} = x_{ij} & \forall (i,j) \in A & \quad (16)\\
& y_{ijk} \ge 0 & (i,j),(j,k) \in A, i \ne k & \quad (17)\\
& x \in \mathcal{X}, \ x \in \{0,1\}. &&
\end{aligned}$$

Notice that problem LP1 was proposed in [6] without applying the RLT. In practice, it can be obtained using a partial application of the RLT to the AQTSP as follows. First, we multiply equations (1), for each $i \in V$, by each of the $(n-1)$ variables x_{ki} for $k \ne i$. Then, for each $j \in \{1,\dots,n\}$, we multiply equations (2) by each of the $(n-1)$ variables x_{jk} for $k \ne j$. All $2n(n-1)$ such quadratic equations are included within the formulation. We then set $x_{ij}x_{ji} = 0$ for all $(i,j) \in A$ with $i \ne j$. The linearization step makes the substitution of $y_{ijk} = x_{ij}x_{jk}$ for all $(i,j),(j,k)$ with $k \ne i$ and enforces that all resulting variables y_{ijk} are nonnegative, to rewrite the problem as the mixed 0-1 linear program LP1.

Using the compact formulation of the AQTSP discussed in Section 2, we define a compact formulation of LP1, called CLP1 as follows:

$$\text{CLP1:} \quad \min \quad \left\{ \sum_{(\langle i,j\rangle,\langle j,k\rangle)\in\mathcal{A}} Q_{ijk} y_{ijk} \ : (15)-(17), x \in \mathcal{X}' \right\}.$$

Padberg and Sung [18] showed that the LP relaxation of the MTZ formulation yields an extremely weak lower bound for the ATSP. However, our computational results show that this is not true for the AQTSP.

4. A column generation approach

In this section we develop a column generation approach to solve the cycle based formulations D1 and D2 presented in section 3.1. Since the number of cycles in $\mathcal{C}$ is exponential with respect to the number of vertices, we first consider a master problem with a feasible subset of cycles, $\bar{\mathcal{C}} \subseteq \mathcal{C}$. Note that the subset $\bar{\mathcal{C}}$ for problem D1 must include at least one tour, while an initial set $\bar{\mathcal{C}}$ for problem D2 must satisfy constraints (11). Let us first start with problem D1 and suppose that $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ and z are the dual variables corresponding to constraints (8) and (9), respectively. The reduced cost of the variable λ_C for each $C \in \mathcal{C}$ is

$$\overline{q(C)} = q(C) - \pi(V \setminus C) - z.$$

A column entering the basis can be found by computing a minimum cost cycle with respect to $Q_{ijk} + \pi_j$ for each $(i,j) \in E, (j,k) \in E$. Let $\overline{q(C^p)} = \min_{C \in \mathcal{C}} \overline{q(C)}$. If $\overline{q(C^p)} \geq 0$ then the current solution is optimal. Otherwise we select column C^p to enter the basis.

Theorem 1. *If the column C^p entering the basis corresponds to a tour, then it is an optimal tour.*

Proof. Consider any cycle C. Since column C^p is the selected column to enter the basis we have

$$\overline{q(C^p)} = q(C^p) - \pi(V \setminus C^p) - z \leq q(C) - \pi(V \setminus C) - z.$$

If cycle C is also a tour, then $q(C^p) \leq q(C)$. □ □

Note that for problem D2, the reduced cost of the variable λ_C for each $C \in \mathcal{C}$ is modified to:

$$\overline{q(C)} = q(C) - \pi(C).$$

4.1. Pricing subproblems

As is clear from the previous section, a column entering the basis can be found by computing a minimum cost cycle in the original graph G with respect to $Q_{ijk} + \pi_j$. Looking for a cycle having minimum negative cost with respect to a quadratic objective is itself an interesting combinatorial optimization problem, which is NP-hard [3, 19]. In this section we explain how to update the linearized models, presented in Section 2, to solve the pricing problems.

Consider LP1 formulation. We first present a revised formulation of the problem similar to the one proposed in [20] for the RLT representation of the Quadratic Assignment Problem. The new formulation, which is obtained by removing constraints (1) from the LP1 formulation, has a graph representation as follows. Consider a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$. For each arc $(i,j) \in A$ we create a node $\langle i,j \rangle$, and for each pair of arcs $(i,j), (j,k)$ with $i \neq k$, we introduce arc $(\langle i,j \rangle, \langle j,k \rangle)$ with weight $w_{ijk} = Q_{ijk}$. If we partition the set of nodes of $\mathcal{G}$ into n clusters $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_n$ such that $\mathcal{V}_i = \{\langle i,j \rangle : j = 1, 2, \ldots, n, j \neq i\}$ for $i = 1, 2, \ldots, n$, then all arcs in the original graph are defined between nodes $\langle i,j \rangle$ and $\langle j,k \rangle$ from different clusters such that $Q_{ijk} > 0$; therefore there are no intra-set arcs. Figure 1 represents the

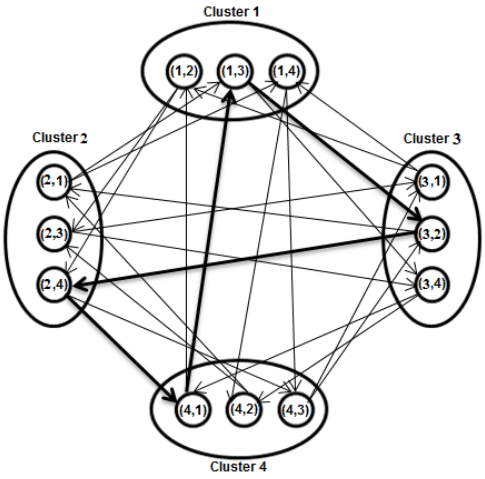

Figure 1: Extended graph $\mathcal{K}_4$ of the complete graph K_4

extended graph, $\mathcal{K}_4$, of a directed complete graph, K_4. The bold lines illustrate a 4-arc cycle that goes through each cluster exactly once.

It is worth to mention that a similar transformation has been proposed in [1]. However, in the transformation of [1] infeasible arcs are also created, though penalized by big costs. More precisely they define a cost function c and set $c_{ijkl} = Q_{ijk}$ for all $(i,j),(k,l) \in A$ with $j = k, i \neq l$, otherwise they set $c_{ijkl} = 2M$, where M is big constant.

The following lemma is a straightforward result implied by construction of $\mathcal{G}$.

Lemma 2. *For any tour $T \in \mathcal{T}$ in G there exists an n-arc cycle $\bar{\bar{T}}$ in $\mathcal{G}$ that goes through each cluster exactly once with $q(T) = w(\bar{T})$ and vice versa.*

Now consider the problem of finding a minimum cost n-arc cycle on graph $\mathcal{G}$ which includes exactly one node from each cluster. This problem is called *Asymmetric Generalized* TSP (AGTSP) and is NP-hard, as it can be reduced to an Asymmetric TSP [21, 22]. The following theorem formally relates the problems AQTSP and AGTSP.

Theorem 3. *An optimal solution for AQTSP can be obtained by solving an AGTSP on $\mathcal{G}$.*

Proof. Let $\bar{T}^*$ be the optimal solution of AGTSP on $\mathcal{G}$. According to Lemma 2 there exists a tour T^* in G with $q(T^*) = w(\bar{T}^*)$. If T^* is not optimal for the AQTSP then there exists an optimal tour T^{**} in G such that $q(T^{**}) < q(T^*)$. From Lemma 2 there must exist an n-arc cycle $\bar{T}^{**}$ in $\mathcal{G}$ with $w(\bar{T}^{**}) = C(T^{**})$. This implies that $w(\bar{T}^{**}) = q(T^{**}) < q(T^*) = w(\bar{T}^*)$ which contradicts the assumption of optimality of $\bar{T}^*$ in $\mathcal{G}$. □ □

Now we turn our attention to the integer linear programming formulation for the AGTSP on $\mathcal{G}$. Defining variables y_{ijk} to indicate whether arc $(\langle i,j\rangle, \langle j,k\rangle)$ is selected or not in the optimal solution, the problem can be stated as follows:

$$\text{AGTSP:} \quad \min \sum_{(\langle i,j\rangle,\langle j,k\rangle)\in\mathcal{A}} Q_{ijk}y_{ijk}$$

$$\text{s.t.} \sum_{j\in V} \sum_{\substack{k\in V:\\ (\langle i,j\rangle,\langle j,k\rangle)\in\mathcal{A}}} y_{ijk} = 1 \qquad \forall i \in V \qquad (18)$$

$$\sum_{k\in V} \sum_{\substack{j\in V:\\ (\langle k,i\rangle,\langle i,j\rangle)\in\mathcal{A}}} y_{kij} = 1 \qquad \forall i \in V \qquad (19)$$

$$\sum_{\substack{k\in V:\\ \langle j,k\rangle\in\mathcal{V}}} y_{ijk} - \sum_{\substack{k\in V:\\ \langle k,i\rangle\in\mathcal{V}}} y_{kij} = 0 \qquad \forall \langle i,j\rangle \in \mathcal{V} \qquad (20)$$

$$\sum_{i\in S} \sum_{\substack{j\notin S:\\ \langle i,j\rangle\in\mathcal{V}_i}} \sum_{\substack{k\in V:\\ \langle j,k\rangle\in\mathcal{V}_j}} y_{ijk} \geq 1 \qquad S \subset V \qquad (21)$$

$$y_{ijk} \in \{0,1\} \qquad \forall (\langle i,j\rangle,\langle j,k\rangle) \in \mathcal{A}. \qquad (22)$$

Constraints (18) require a solution to include exactly one of the arcs entering a cluster, while constraints (19) require a solution to include exactly one of the arcs leaving a cluster. Constraint (20) is equivalent to network flow conservation constraints and ensure that a solution tour is uninterrupted and continuous. Finally, constraints (21) eliminate all subtours.

Theorem 4. *Constraints* (18) *in problem AGTSP are redundant and can be removed from the model.*

Proof. Suppose y^* is a feasible solution for problem AGTSP after removing constraint (18) and define, for each $i \in V$,

$$\epsilon_i = \sum_{j\in V} \sum_{\substack{k\in V:\\ (\langle i,j\rangle,\langle j,k\rangle)\in\overline{E}}} y^*_{ijk}.$$

We show that $\epsilon_i = 1$ for all $i \in V$.

Consider cluster $\mathcal{V}_s$. Then constraint (19) is satisfied for cluster $\mathcal{V}_s$ by u^*, i.e.,

$$\sum_{k\in V} \sum_{\substack{j\in V\\ (\langle k,s\rangle,\langle s,j\rangle)\in\overline{E}}} y^*_{ksj} = 1.$$

Therefore there exists a node $\langle s,t\rangle \in \mathcal{V}_s$ with in-degree one in the tour, while the in-degree of all other nodes in cluster $\mathcal{V}_s$ is zero. Hence by (20) the out-degree of node $\langle s,t\rangle$ must be equal to one while the out-degree of all the other nodes in the same cluster must be zero. □ □

As a consequence of Theorem 4, AGTSP simplifies as follows:

$$\text{AGTSP':} \quad \min \left\{ \sum_{(\langle i,j\rangle,\langle j,k\rangle)\in\mathcal{A}} Q_{ijk} y_{ijk} \;:\; (19)-(22) \right\}.$$

Now let us consider the pricing problem of D1. As we explained above, each feasible tour in G corresponds to a tour in $\mathcal{G}$ that goes through each cluster exactly once. Therefore, finding a negative reduced cost directed cycle for problem D1 is equivalent to solving a modified version of problem AGTSP' which visit each cluster at most once. By defining a binary variable w_i to indicate whether the Cluster $\mathcal{V}_i$ is on the cycle or not, the minimum negative cost cycle is then found by solving the following problem:

$$\begin{aligned}
\text{Sub:} \quad \min \quad & \sum_{(\langle i,j\rangle,\langle j,k\rangle)\in\mathcal{A}} Q_{ijk} y_{ijk} - \sum_{i\in V} \pi_i (1-w_i) - z \\
& \sum_{k\in V} \sum_{\substack{j\in V: \\ (\langle k,i\rangle,\langle i,j\rangle)\in\mathcal{A}}} y_{kij} = w_i \qquad \forall i \in V \qquad (23) \\
& (20)-(22) \\
& w_i \in \{0,1\}.
\end{aligned}$$

Considering the definition of the w_i variables, one can obtain an equivalent formulation for the pricing problem by replacing constraint (23) with the so-called resource constraint (24) and removing variables w_i from the model:

$$\sum_{\substack{j,k\in V \\ (\langle k,i\rangle,\langle i,j\rangle)\in\mathcal{A}}} y_{jik} \leq 1 \quad \forall i \in V. \qquad (24)$$

It should be observed that by forcing the solution of the pricing problem to be a cycle with negative cost one can easily remove constraints (21) from the pricing model. Therefore, finding a cycle with negative reduced cost is simply a matter of finding a path originating and ending at each node $\langle i,j\rangle$ of graph $\overline{G}$ having a negative cost and satisfying the resource constraint (24). Since dual variables π are defined on each cluster $\mathcal{V}_i$, the problem of finding a negative reduced cost cycle can be formulated as resource-constrained elementary shortest path problem. Let $\bar{q}_{st}$ denote the cost of the resource-constrained elementary shortest path from origin node $\langle s,t\rangle$ to itself in $\overline{G}$. The pricing problem can be written as:

$$\min_{\langle s,t\rangle\in\hat{V}} \{q_{st}\}.$$

The solution of the subproblems provides either a certificate of optimality of the current solution (λ,π,z) or a new column C^p that will be added to the master problem. It is worth pointing out that solving the pricing subproblem to optimality is only needed to prove optimality of the current primal and dual solutions; one can stop solving the subproblem whenever a negative reduced cost column is found [23]. This happens because adding this column to $\bar{\mathcal{C}}$ ensures that the new dual solution (π,z) will be different, and therefore the termination of the algorithm.

4.2. Stabilized column generation

Column generation methods usually suffer from slow convergence to the optimal solution and tailing off effects. Primal degeneracy, dual degeneracy and instability in the behavior of dual variables are well known to be the main causes of this behavior [24, 25].

To control the dual variables during the solution process, we use the stabilized column generation approach proposed in [26]. This approach combines the box step method [27] with a kind of descent method proposed in [28]. The box step method introduces a box around the previous dual vector and modifies the master problem such that the feasible dual space is limited to the area defined by these boxes, while the latter tries to adapt the master problem so that the distance separating a dual solution from the previous optimal dual solution is linearly penalized.

In order to present the idea, let us rewrite the restricted version of the master problem D1, for $\bar{\mathcal{C}} \subseteq \mathcal{C}$, as the following model:

$$\begin{aligned} \text{RD1: min} \quad & \sum_{C \in \bar{\mathcal{C}}} q(C)\lambda_C & & \\ \text{s.t.} \quad & \sum_{C \in \bar{\mathcal{C}}: i \in C} \lambda_C \geq 1 & \forall i \in V & \qquad (25) \\ & (9), (10). & & \end{aligned}$$

Note that since the set partitioning constraints admit negative dual values which can be problematic for the sub-problem, we used a relaxed version of the problem as the first step of the stabilization approach.

Consider the dual variables π associated with the constraints (25) and bound each π_i in the interval $[\delta_i^-, \delta_i^+]$. These bounds are first given as parameters to the model and then automatically updated during the process. The dual variable π_i can take values outside the given bounds, but the dual objective is then penalized by $\varepsilon_i^-(\delta_i^- - \pi_i)$ if $\pi_i < \delta_i^-$ and by $\varepsilon_i^+(\delta_i^+ - \pi_i)$ if $\pi_i > \delta_i^+$. The dual of the problem RD1 then becomes:

$$\begin{aligned} \text{SP: max} \quad & z + \sum_{i \in V} \pi_i - \varepsilon_i^- w_i^- - \varepsilon_i^+ w_i^+ & & \\ \text{s.t.} \quad & z + \sum_{i \in C} \pi_i \leq q(C) & \forall C \in \bar{\mathcal{C}} & \\ & \pi_i + w_i^- \geq \delta_i^- & \forall i \in V & \\ & \pi_i - w_i^+ \leq \delta_i^+ & \forall i \in V & \\ & \pi, w^-, w^+ \geq 0, \ z \text{ unrestricted.} & & \end{aligned}$$

The primal of the stabilized restricted master problem, and hence the dual of SP, is:

$$
\begin{aligned}
\text{SD1: min} \quad & \sum_{C\in\bar{\mathcal{C}}} q(C)\lambda_C + \sum_{i\in V} -\delta_i^-\mu_i^- + \delta_i^+\mu_i^+ & \\
\text{s.t.} \quad & \sum_{C\in\bar{\mathcal{C}}: i\in C} \lambda_C - \mu_i^- + \mu_i^+ \geq 1 & \forall i \in V \\
& \sum_{C\in\bar{\mathcal{C}}} \lambda_C = 1 & \\
& \mu_i^- \leq \varepsilon_i^- & \forall i \in V \\
& \mu_i^+ \leq \varepsilon_i^+ & \forall i \in V \\
& \lambda, \mu^-, \mu^+ \geq 0. &
\end{aligned}
$$

This method is referred to as BoxPen stabilization since the bounds (δ^-, δ^+) on the original dual variables π can be represented by a bounding box containing the current dual solution. Note that the stabilized version of the problem D2 is the same as SD1 without the convexity constraint and is called SD2. In order to use the stabilized models efficiently, one must initialize and update the parameters correctly. In order to reduce the dual variables' variations, select $[\delta^-, \delta^+]$ to form a small box containing the current dual solution, and solve the problem SD1 (SD2). At the first iteration, when no solution is available, the dual variables π can be simply estimated. If the new π lies in the box $[\delta^-, \delta^+]$, reduce its width and augment the penalty given by ε^- and ε^+. Otherwise, enlarge the box and decrease the penalty. The update could be performed in each iteration, or alternatively, each time a dual solution of currently best value is obtained. In Section 5 we discuss more about the updating process.

5. Computational experiments

In this section we present our computational experiments on the lower bound computations for the AQTSP. We used the AMPL modeling language [29] with Cplex 12.0.1 as linear solver for the RMP and as a mixed integer linear solver for the pricing problem on an Intel Core i5-2410M CPU with 2.30 GHz and 6 GB RAM in single processor mode. We implemented the algorithms for solving the RLT and the LP models in C++ language. In the following we present the test instances, the parameter setting of the stabilized column generation approach, and provide the comparison of the lower bounding approaches in details.

5.1. Test instances

We considered six different benchmark sets introduced in [1].

- RandomClass: It consists of a class of randomly generated instances with size ranging from $n = 5$ to $n = 25$. All instances are complete graphs with n vertices and $m = n(n-1)/2$ edges. The quadratic cost Q_{ijk} for $(i,j),(j,k) \in A$ with $i \neq k$ is defined as an integer number uniformly chosen within the set $\{0, 1, ..., 10000\}$.

Table 1: Comparison of CG and SCG approaches in terms of CPU execution times on RandomClass data set.

Size	n=10				n=20			
	CG2	SCG2	CG1	SCG1	CG2	SCG2	CG1	SCG1
	3.9	3.2	8.9	7.3	97.4	34.9	508.5	271.8

- ReloadClass: The random graphs $G(V, A)$ with different number of nodes and arcs are generated as follows: For each V with $n \in \{15, 20\}$ arc (i, j), $i, j \in V, i \neq j$ is present with probability $p \in \{\frac{1}{2}, 1\}$. If an arc (i, j) is present, then it is colored randomly with one of the colors in $D = \{1, \ldots, d\}$ where $d \in \{5, 10, 20\}$. Based on the cost c_{ts} of changing from color $t \in D$ to color $s \in D$, the following data set is considered:
  - ReloadClass1: For all $(i, j), (j, k) \in A$ with colors t and s, respectively, $Q_{ijk} = c_{ts} = 1$.
  - ReloadClass2: For all $(i, j), (j, k) \in A$ with colors t and s, respectively, $Q_{ijk} = c_{ts}$ where c_{st} chosen uniformly at random in $\{1, \ldots, 10\}$.
- RealClass1, RealClass2, RealClass3: Real instances, with $n \in \{6, \ldots, 41\}$, arising in a computational biology applications [1, 2].

5.2. Stabilization parameter choice

We implemented the stabilized column generation approach using different sets of initial values. In the following we present the results of some preliminary experiments whose purpose was to initialize and update the parameters for both SD1 and SD2.

For the problem SD1, we initialized δ^- and δ^+ at -1000 and 1000 respectively. The vector parameter ε^- and ε^+ were selected as -5 and 5 respectively and were kept fixed throughout the solution process. We updated the parameter (δ^-, δ^+) from $(-1000, 1000)$ to $(\tilde{\pi} - 100, \tilde{\pi} + 100)$, where $\tilde{\pi}$ is the current dual solution, only if the column returned by the subproblem had a non-negative reduced cost and $(\mu^-, \mu^+) \neq (0, 0)$. The stopping criteria of the stabilized column generation algorithm is $\overline{r(C)} \geq 0$ and $(\mu^-, \mu^+) = (0, 0)$.

In order to find potentially good initial values of (δ^-, δ^+) for problem SD2, we first solved the problem D2 with a feasible subset of cycles. By using the dual variables $\tilde{\pi}$ of problem D2, we initialized (δ^-, δ^+) with $(\tilde{\pi} - 10, \tilde{\pi} + 10)$. The vector parameter ε^- and ε^+ were initially set to 0.0001. If the subproblem is able to find a negative reduced cost cycle, then the values of ε^- and ε^+ were increased by 10%. However, when there was no more such column and $(\mu^-, \mu^+) \neq (0, 0)$, the values of ε^- and ε^+ were decreased by dividing each one by 100 and the parameter (δ^-, δ^+) were updated to $(\tilde{\pi} - 100, \tilde{\pi} + 100)$, where $\tilde{\pi}$ is the current dual solution of the SD2. The stopping criterion of the stabilized column generation algorithm is the same as that of problem SD1.

Table 2: Comparison of different lower bounding approaches on RandomClass data set.

Instance		Gap(%)				CPU Time				CutGap(%)	
n	Opt.	CLP1	RLT1	SCG2	SCG1	CLP1	RLT1	SCG2	SCG1	LP1	LP1$^+$
5	12373.2	13.1	0.0	10.1	0.0	0.0	0.0	1.8	2.1	0.0	0.0
6	11883.1	13.4	10.1	10.8	10.1	0.0	0.0	1.8	1.5	0.0	0.0
7	13204.9	20.6	4.2	13.9	3.5	0.0	0.6	2.0	3.0	5.2	0.9
8	13363.4	24.3	9.2	16.2	3.6	0.0	1.9	2.1	3.2	9.0	4.7
9	13063.2	26.4	15.7	19.3	7.1	0.0	3.3	2.5	5.2	18.9	12.5
10	12921.5	27.0	19.5	19.0	3.3	0.0	7.8	3.3	7.4	20.9	14.3
11	12997.5	31.0	26.2	23.9	7.0	0.0	11.5	4.4	12.0	27.3	23.4
12	11629.5	25.0	21.7	16.7	6.3	0.0	15.7	5.8	10.9	20.8	19.3
13	12679.5	30.8	29.3	21.1	4.8	0.0	26.0	8.6	21.1	28.3	26.0
14	11838.3	30.9	29.5	23.9	10.2	0.0	35.1	10.0	20.7	29.2	25.8
15	12428.8	34.9	34.7	23.0	5.5	0.0	48.1	16.8	37.1	34.8	31.0
16	12135.7	36.3	36.3	27.8	8.4	0.0	81.4	16.6	47.3	36.5	34.0
17	11832.2	38.8	39.7	26.6	14.7	0.0	81.3	21.7	60.0	38.8	34.4
18	11662.2	36.7	37.9	26.7	4.8	0.0	109.2	24.4	119.2	36.7	33.2
19	12095.9	39.9	41.2	29.2	7.3	0.0	146.2	31.2	157.5	39.5	36.5
20	11802.8	41.1	42.9	30.5	6.2	0.0	182.6	34.9	271.8	41.0	37.1
21	11288.2	38.4	41.3	27.9	4.2	1.0	247.0	51.4	435.2	38.6	36.3
22	11741.0	41.0	44.6	32.3	10.4	1.0	317.7	51.6	480.2	40.8	38.0
23	11549.7	41.2	44.6	33.4	8.7	1.0	399.7	59.1	671.2	41.2	38.4
24	11239.1	41.5	46.0	31.3	9.4	1.0	503.2	84.0	877.8	41.4	39.7
25	11434.8	41.7	46.1	31.9	8.9	1.0	659.5	105.4	1698.3	41.7	40.3

In order to show the effectiveness of stabilization, we compare the computational time of column generation to its stabilized version on two instances of different dimensions in Table 1. Each row of the table reports the average computational time over ten instances of the same size. CG1 and CG2 stand for Column Generation approach to the problem SD1 and SD2 respectively. SCG1 and SCG2 are for the stabilized version of the CG1 and CG2 respectively. The results show that stabilization is effective for both symmetric and asymmetric instances.

5.3. Lower bound computation

As we proved in Theorem 4, if the column entering the basis corresponds to a tour, then it is an optimal tour. Therefore, in our CG approach, a solution of the pricing problem, which may contain a single cycle or multiple ones, is optimal for the master problem RD1 if it covers all the nodes $i \in V$. Since looking for a single cycle in the pricing problem requires some kind of subtour elimination constraint, we restrict the search to find a cycle (single or multiple) with negative cost. Therefore, when no more new columns can be priced out, a solution of the master problem RD1 gives a lower bound for the original problem. Note that the optimal value of the problem RD2 always gives a lower bound for the original problem, regardless of the solution being a single cycle or multiple ones.

In Tables 2 to 6 we present computational results of the lower bounding schemes for

Table 3: Comparison of different lower bounding approaches on ReloadClass1 and ReloadClass2 data set.

Instance			Gap(%)				CPU Time				CutGap(%)	
n	(p,c)	Opt.	CLP1	RLT1	SCG2	SCG1	CLP1	RLT1	SCG2	SCG1	LP1	LP1$^+$
15	(5,5)	4.1	41.5	29.3	56.1	12.2	<1	22.9	5.1	1.5	25.6	12.2
15	(5,10)	6.5	20.0	13.8	20.0	9.2	<1	24.3	12.5	1.4	9.1	6.2
15	(5,20)	8.5	11.8	7.1	10.6	7.1	<1	27.5	18.9	1.4	3.1	0.0
15	(10,5)	0.4	100.0	100.0	100.0	100.0	<1	27.2	4.6	1.5	100.0	100.0
15	(10,10)	2.9	89.7	86.2	86.2	51.7	<1	27.1	4.9	1.8	79.7	55.2
15	(10,20)	5.1	29.4	27.5	33.3	9.8	<1	32.0	12.9	1.8	22.5	15.7
Ave.		4.5	48.7	44.0	51.0	31.7		25.8	9.2	1.5	40.0	31.5
20	(5,5)	3.2	71.9	71.9	81.3	40.6	<1	49.9	6.1	1.8	61.6	54.7
20	(5,10)	6.1	29.5	26.2	39.3	8.2	<1	57.3	12.8	1.9	24.3	18.0
20	(5,20)	9.3	16.1	12.9	17.2	5.4	<1	58.5	31.1	1.7	8.9	3.8
20	(10,5)	0.0	0.0	0.0	0.0	0.0	<1	68.5	6.8	1.9	0.0	0.0
20	(10,10)	2.4	100.0	100.0	100.0	91.7	<1	68.7	6.0	2.6	100.0	97.5
20	(10,20)	4.8	58.3	60.4	62.5	27.1	<1	66.7	7.4	2.8	56.5	41.9
Ave.		4.3	46.0	45.2	50.0	28.8		61.6	11.7	2.1	41.9	36.0
15	(5,5)	17.7	41.2	25.4	62.1	11.9	<1	24.7	5.6	1.5	30.5	22.1
15	(5,10)	23.3	26.6	58.8	27.5	8.6	<1	25.2	11.7	1.5	21.3	17.7
15	(5,20)	27.2	23.9	26.5	23.2	9.2	<1	23.3	12.7	1.4	18.1	12.0
15	(10,5)	2.1	100.0	100.0	100.0	100.0	<1	26.2	4.6	1.6	100.0	100.0
15	(10,10)	6.5	81.5	80.0	100.0	50.8	<1	27.0	5.0	1.8	79.1	59.8
15	(10,20)	11.7	47.9	45.3	47.9	17.1	<1	31.4	8.0	2.0	46.3	34.3
Ave.		14.8	53.5	56.0	60.1	32.9		26.3	7.9	1.6	49.2	41.0
20	(5,5)	8.3	69.9	68.7	89.2	25.3	<1	57.6	5.5	2.0	57.7	50.8
20	(5,10)	19.2	40.1	43.2	52.1	14.1	<1	68.4	9.4	1.9	33.0	25.2
20	(5,20)	26.8	25.4	51.5	23.1	7.5	<1	65.4	20.4	1.9	22.6	19.0
20	(10,5)	0.0	0.0	0.0	0.0	0.0	<1	74.7	6.2	2.1	0.0	0.0
20	(10,10)	4.5	100.0	100.0	100.0	100.0	<1	74.5	5.6	2.5	100.0	98.1
20	(10,20)	9.8	59.2	66.3	74.5	21.4	<1	73.9	7.7	2.9	58.8	44.5
Ave.		11.4	49.1	55.0	56.5	28.0		69.1	9.1	2.2	45.3	39.6

different test instances. We compare different lower bounding procedures proposed in the paper in terms of average gap and average computational time. The formula we used to compute the percent gaps is $100 \times (\text{Opt} - \text{Lb})/\text{Lb}$, where Lb stands for the value of the lower bound. Each row of the tables reports the average results over ten instances of the same size. Morover, we compare the bounding procedure proposed in the paper with bounds presented in [6] in terms of average gap. The first one is the bound of LP1 formulation obtained by dynamically separating the subtour elimination constraints (3) by means of the LEMON Graph Library [30], while the second one LP1$^+$ considers additional specific valid inequalities to LP1. Since these results have been provided directly by the author [6], computational times are not available.

Table 2 reports the results for RandomClass instances. The problem size n is found in the first column of the table. The second column shows the average optimal values (Opt.) of the 10 instances for each dimension. The next four columns, from left to right, give the

Table 4: Comparison of different lower bounding approaches on RealClass1 data set.

Instance		Gap(%)				CPU Time				
n	Opt.	CLP1	RLT1	SCG2	SCG1	CLP1	RLT1	SCG2	SCG1	LP1 (Gap(%))
20	14742	14.9	14.9	36.4	13.6	0.0	110.0	15.6	10.2	9.2
21	15514	14.9	14.9	35.7	13.7	0.0	134.0	17.9	11.3	9.8
22	16264	15.1	15.1	36.1	13.7	0.0	156.0	16.3	11.0	11.0
23	16919	14.9	15.1	35.4	13.8	1.0	186.0	15.4	12.8	9.8
24	17660	14.7	14.9	35.3	13.2	1.0	213.0	17.9	13.4	10.8
25	18327	14.6	14.7	35.1	12.9	2.0	247.0	23.7	14.2	12.0
26	19046	14.4	14.8	34.6	13.4	3.0	303.0	25.6	15.2	9.0
27	19687	14.1	14.4	34.6	12.6	3.0	359.0	26.3	22.1	10.6
28	20408	14.0	14.4	33.9	12.3	6.0	427.0	23.3	22.5	12.4
29	21123	13.9	14.5	34.1	12.3	6.0	475.0	28.2	25.4	10.2
30	21785	13.6	15.1	33.6	12.0	6.0	556.0	33.4	31.9	12.1
31	22478	13.6	14.5	33.6	11.8	9.0	632.0	33.8	38.8	12.1
32	23166	13.7	15.7	33.4	11.8	7.0	736.0	38.0	39.7	11.0
33	23894	13.7	15.3	33.5	11.6	9.0	829.0	31.5	46.0	12.3
34	24600	13.8	15.3	33.3	11.7	8.0	899.0	34.5	96.2	10.5
35	25337	13.8	15.9	33.4	11.7	12.0	970.0	37.1	71.6	10.6
36	26109	14.0	15.6	33.4	11.4	19.0	1107.0	48.2	163.7	9.6
37	26854	14.1	16.0	33.7	11.6	17.0	1262.0	56.7	130.8	12.9
38	27604	14.2	16.8	33.6	11.9	19.0	1406.0	50.6	128.9	13.0
39	28389	14.3	17.4	33.9	11.6	26.0	1581.0	50.9	229.3	13.1
40	29109	14.5	17.6	33.8	11.8	22.0	1791.0	62.3	235.6	12.4
41	29844	14.5	18.3	33.9	12.1	24.0	1993.0	55.1	287.5	12.9

percent gaps obtained using the compact formulation CLP1, the dual-ascent implementation of RLT1, the Stabilized Column Generation approach applied to the problem SD2 (SCG2), and the Stabilized Column Generation approach applied to the problem SD1 (SCG1). The CPU execution times of each approach are given in the next four columns of the table. The last two columns of the table report the percent gaps obtained using two lower bounds LP1 and LP1$^+$ presented in [6].

As we can observe from the results in Table 2, SCG1 outperforms the best literature bounds LP1 and LP1$^+$ when the size of the instances grows, and the difference becomes more evident for the largest instances. Also SGC2 has an interesting behavior since it provides bounds that are comparable with LP1 and LP1$^+$, but even better for the largest instances. Though it does not provide a bound as good as SCG1, its computational times are much shorter, especially for large instances. Note also that the results provided by RLT1 are not much better from those of the compact linearized formulation CLP1 when the size is over a certain threshold, and actually in some cases it is even worse. The reasons can be stated as follows: First, the dual-ascent strategy applied to solve the Lagrangian dual problem does not actually calculate the RLT1 bound, but approximates it. In fact the dual heuristic may stopped after reaching a sub-optimal solution. Although we used a simulated

Table 5: Comparison of different lower bounding approaches on RealClass2 data set.

Instance		Gap(%)				CPU Time				
n	Opt.	CLP1	RLT1	SCG2	SCG1	CLP1	RLT1	SCG2	SCG1	LP1 (Gap(%))
20	14703	15.0	15.1	40.9	13.7	0.0	111.0	21.1	11.7	11.4
21	15480	15.0	15.0	40.2	13.7	0.0	132.0	23.2	10.8	9.2
22	16233	15.2	15.3	40.8	13.8	1.0	154.0	26.5	12.6	11.3
23	16892	15.0	15.3	40.1	13.8	1.0	188.0	27.7	15.0	12.1
24	17638	14.7	14.9	40.1	13.3	2.0	218.0	24.6	14.8	9.6
25	18309	14.7	14.9	39.9	13.0	1.0	250.0	24.7	17.7	10.6
26	19033	14.5	14.9	39.5	13.4	4.0	297.0	30.8	15.8	11.3
27	19677	14.2	14.5	39.5	12.6	3.0	354.0	33.8	25.0	10.3
28	20402	14.1	14.8	38.9	12.4	4.0	409.0	40.2	23.8	12.4
29	21121	14.0	14.3	39.1	12.3	6.0	462.0	38.9	28.7	9.9
30	21793	13.7	15.1	38.7	12.0	7.0	550.0	42.8	33.4	12.2
31	22485	13.7	15.2	38.8	11.8	6.0	616.0	40.6	35.1	9.7
32	23177	13.7	15.8	38.6	11.9	8.0	719.0	50.7	39.6	11.4
33	23909	13.8	15.7	38.7	11.6	10.0	805.0	65.3	40.9	10.3
34	24619	13.8	16.1	38.6	11.7	10.0	922.0	64.4	69.2	10.0
35	25360	13.8	16.1	38.7	11.7	9.0	1029.0	66.4	121.8	9.6
36	26136	14.0	15.8	38.7	11.4	15.0	1183.0	78.8	133.2	9.6
37	26885	14.2	16.5	39.0	11.6	16.0	1270.0	75.1	125.0	12.9
38	27640	14.2	17.1	38.9	11.9	16.0	1423.0	75.9	191.7	11.8
39	28429	14.4	16.4	39.2	11.6	19.0	1655.0	90.4	194.1	11.9
40	29153	14.5	18.2	39.2	11.8	20.0	1819.0	96.0	124.4	12.4
41	29891	14.6	18.1	39.3	12.1	24.0	2002.0	110.1	198.1	12.2

annealing step to shake the dual-ascent bound calculation out of local optima as proposed in [16], it was not efficient. The second and most important reason is in using the Gilmore-Lawler type procedure to solve the subproblem (13). When solving a subproblem for an arcs (k, l) (an ATSP containing a fixed arc (k, l)) the optimal solution $\bar{\boldsymbol{y}}$ of subproblem will be implied by an ATSP in which only the cost of two arcs adjacent to (k, l) is considered, namely the arcs (i, k) and (l, j) with the smallest costs Q_{ikl} and Q_{klj}. This fact suggests how underestimated the costs D_{kl} may be for the computation of the Gilmore-Lawler lower bound. Another interesting result observed from Table 2 is the quality of bounds obtained using the compact formulation CLP1. The results indicate that the duality gaps implied by the LP1 formulation of Fischer [6] and the compact formulation CLP1 are competitive (especially when the dimension of the problem is increased). This shows that the main difficulty of the AQTSP lies in the quadratic structure of the objective function rather than the TSP structure.

Table 3 presents the computational results for the ReloadClass1 and ReloadClass2 instances. The first three columns indicate the problem size (n), the probability and number of colors (p, c), and the average optimal values (Opt.) of 10 instances for each dimension. The next four columns, from left to right, give the percent gaps obtained by CLP1, RLT1,

Table 6: Comparison of different lower bounding approaches on RealClass3 data set.

Instance		Gap(%)				CPU Time				
n	Opt.	CLP1	RLT1	SCG2	SCG1	CLP1	RLT1	SCG2	SCG1	LP1 (Gap(%))
20	14597	14.6	14.7	46.2	13.3	1.0	114.0	17.8	12.0	11.7
21	15391	14.6	14.8	45.9	13.1	1.0	134.0	17.6	12.9	11.5
22	16163	14.8	14.9	46.4	13.4	0.0	154.0	17.9	13.0	10.8
23	16840	14.6	14.8	46.0	13.4	1.0	187.0	23.5	13.3	11.1
24	17605	14.3	14.5	46.0	12.8	2.0	213.0	21.9	15.4	12.5
25	18296	14.3	14.5	46.1	12.6	2.0	255.0	24.3	15.8	9.9
26	19039	14.1	14.5	45.5	13.0	4.0	305.0	24.7	19.8	11.2
27	19704	13.7	14.2	45.9	12.1	2.0	352.0	27.7	18.4	11.3
28	20449	13.7	14.2	45.4	12.0	5.0	415.0	30.3	24.9	10.6
29	21190	13.6	14.0	45.6	11.9	4.0	471.0	31.5	27.0	10.2
30	21883	13.3	14.7	45.3	11.6	6.0	545.0	34.4	26.9	11.9
31	22596	13.2	14.4	45.5	11.4	7.0	618.0	34.9	36.1	11.8
32	23310	13.3	14.5	45.3	11.4	9.0	732.0	36.0	34.0	10.1
33	24064	13.4	15.0	45.5	11.2	9.0	825.0	46.8	41.4	9.9
34	24796	13.4	15.2	45.5	11.3	9.0	906.0	52.4	44.1	10.2
35	25559	13.4	15.2	45.6	11.3	14.0	990.0	53.7	85.0	11.6
36	26358	13.6	15.3	45.7	11.0	15.0	1114.0	61.1	69.5	10.9
37	27129	13.7	15.8	46.0	11.2	12.0	1282.0	63.0	106.0	12.1
38	27906	13.8	16.0	46.0	11.5	22.0	1409.0	61.4	194.2	11.2
39	28718	13.9	16.2	46.3	11.2	20.0	1606.0	63.9	189.1	12.8
40	29465	14.1	16.4	46.3	11.4	25.0	1784.0	92.1	213.6	13.0
41	30227	14.1	16.5	46.4	11.7	28.0	1960.0	88.7	276.9	13.1

SCG2, and SCG1. The CPU execution times of each approach are given in the next four columns of the table. The last two columns report the best known literature lower bounds LP1 and LP1$^+$ for the ReloadClass presented in [6]. The results in this table confirm the indications arising from the random instances, namely that SCG1, with the exception of few classes of instances, is much better than all the other bounds, however this time the computational times of SCG2 are not so competitive. Also in this case the results of RLT1 appear unattractive, but as in random instances the gaps implied by compact formulation CLP1 are competitive with those of LP1.

Tables 4 to 6 present the computational results for RealClass1, RealClass2 and RealClass3 instances, respectively. The structure of the tables is the same as Table 2. For these data sets, the LP1$^+$ bounds are not available. In this case we can observe that the LP1 bound is tighter with respect to the previous classes of instances. The SCG1 is comparable with LP1 (especially when the size of the instances grows), but SCG2 is the worst. Considering the CPU times, the RLT1 results are still unattractive, while the CLP1 provides the best results.

6. Conclusions

In this paper we studied different lower bounding scheme for AQTSP. We proposed a cycle formulation for the problem and solved the resulting LP problem by a Column Generation approach. We presented a linearized formulation proposed in the literature and developed some reformulations to it. Moreover, we introduced a mixed 0-1 linear programming formulation based on the reformulation-linearization technique and developed a Lagrangian relaxation algorithm based on it. We have shown how the linearized formulations can be applied to finding the negative reduced cost in the pricing problem of the Column Generation approach. To overcome the problems of instability in the behavior of dual variables of the presented master problem, we used a stabilized column generation approach. The specific main findings of the work presented in computational experiments are very interesting. First, the linearized formulation of AQTSP combined with compact formulation (MTZ) of Miller, Tucker and Zemlin [10] for the ATSP provides the bounds which (in contrast to ATSP) are competitive to those obtained by the linearized formulation combined with the Dantzig-Fulkerson-Johnson formulation [9]. Second, the dual-ascent algorithm applied to RLT representation of AQTSP (in contrast to QAP) does not provide a good approximation of the optimal solution of the problem. Therefore, the lower bound obtained by a RLT representation of AQTSP is unattractive. And finally, our column generation approach is very promising as it outperforms the others in almost all the test instances.

7. Acknowledgments

The first author has been supported by the German Research Foundation (DFG) under grant BU 2313/2.

References

[1] A. Fischer, F. Fischer, G. Jger, J. Keilwagen, P. Molitor, I. Grosse, Exact algorithms and heuristics for the quadratic traveling salesman problem with an application in bioinformatics, Discrete Applied Mathematics 166 (0) (2014) 97 – 114.

[2] G. Jäger, P. Molitor, Algorithms and experimental study for the traveling salesman problem of second order, in: Combinatorial Optimization and Applications, Lecture Notes in Computer Science, 5165, Springer, 2008, pp. 211–224.

[3] A. Aggarwal, D. Coppersmith, S. Khanna, R. Motwani, B. Schieber, The angular-metric traveling salesman problem, SIAM Journal on Computing 29 (3) (2000) 697–711.

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

[5] A. Fischer, C. Helmberg, The symmetric quadratic traveling salesman problem, Mathematical Programming 142 (1-2) (2013) 205–254.

[6] A. Fischer, An analysis of the asymmetric quadratic traveling salesman polytope, SIAM Journal on Discrete Mathematics 28 (1) (2014) 240–276. arXiv:http://dx.doi.org/10.1137/110858665.

[7] W. P. Adams, H. D. Sherali, A tight linearization and an algorithm for zero-one quadratic programming problems, Management Science 32 (10) (1986) 1274–1290.

[8] W. P. Adams, H. D. Sherali, Linearization strategies for a class of zero-one mixed integer programming problems, Operations Research 38 (2) (1990) 217–226.

[9] G. Dantzig, R. Fulkerson, S. Johnson, Solution of a large-scale traveling-salesman problem, Journal of the operations research society of America 2 (4) (1954) 393–410.
[10] C. E. Miller, A. W. Tucker, R. A. Zemlin, Integer programming formulation of traveling salesman problems, J. ACM 7 (4) (1960) 326–329.
[11] F. Glover, E. Woolsey, Technical noteconverting the 0-1 polynomial programming problem to a 0-1 linear program, Operations Research 22 (1) (1974) 180–182. arXiv:http://dx.doi.org/10.1287/opre.22.1.180.
[12] M. Held, R. M. Karp, The traveling-salesman problem and minimum spanning trees, Operations Research 18 (6) (1970) 1138–1162.
[13] W. P. Adams, T. A. Johnson, Improved linear programming-based lower bounds for the quadratic assignment problem, DIMACS series in discrete mathematics and theoretical computer science 16 (1994) 43–77.
[14] P. C. Gilmore, Optimal and suboptimal algorithms for the quadratic assignment problem, Journal of the Society for Industrial & Applied Mathematics 10 (2) (1962) 305–313.
[15] E. L. Lawler, The quadratic assignment problem, Management science 9 (4) (1963) 586–599.
[16] P. Hahn, T. Grant, Lower bounds for the quadratic assignment problem based upon a dual formulation, Operations Research 46 (6) (1998) 912–922.
[17] W. P. Adams, M. Guignard, P. M. Hahn, W. L. Hightower, A level-2 reformulation-linearization technique bound for the quadratic assignment problem, European Journal of Operational Research 180 (3) (2007) 983 – 996.
[18] M. Padberg, T.-Y. Sung, An analytical comparison of different formulations of the travelling salesman problem, Mathematical Programming 52 (1-3) (1991) 315–357.
[19] G. Galbiati, S. Gualandi, F. Maffioli, On minimum reload cost cycle cover, Discrete Applied Mathematics 164, Part 1 (0) (2014) 112 – 120.
[20] B. Rostami, F. Malucelli, A revised reformulation-linearization technique for the quadratic assignment problem, Discrete Optimization 14 (0) (2014) 97 – 103.
[21] G. Laporte, H. Mercure, Y. Nobert, Generalized traveling salesman problem through n sets of nodes: the asymmetrical case, Discrete Applied Mathematics 18 (2) (1987) 185 – 197.
[22] C. E. Noon, J. C. Bean, A lagrangian based approach for the asymmetric generalized traveling salesman problem, Operations Research 39 (4) (1991) 623–632.
[23] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. Savelsbergh, P. H. Vance, Branch-and-price: Column generation for solving huge integer programs, Operations research 46 (3) (1998) 316–329.
[24] P. C. Gilmore, R. E. Gomory, A linear programming approach to the cutting-stock problem, Operations research 9 (6) (1961) 849–859.
[25] B. Kallehauge, J. Larsen, O. B. Madsen, Lagrangian duality applied to the vehicle routing problem with time windows, Computers & Operations Research 33 (5) (2006) 1464–1487.
[26] O. Du Merle, D. Villeneuve, J. Desrosiers, P. Hansen, Stabilized column generation, Discrete Mathematics 194 (1) (1999) 229–237.
[27] R. Marsten, W. Hogan, J. W. Blankenship, The boxstep method for large-scale optimization, Operations Research 23 (3) (1975) 389–405.
[28] S. Kim, K.-N. Chang, J.-Y. Lee, A descent method with linear programming subproblems for nondifferentiable convex optimization, Mathematical Programming 71 (1) (1995) 17–28.
[29] R. Fourer, D. M. Gay, B. W. Kernighan, AMPL: A mathematical programming language, AT&T Bell Laboratories Murray Hill, NJ 07974, 1987.
[30] B. Dezs, A. Jüttner, P. Kovács, Lemon - an open source c++ graph template library, Electron. Notes Theor. Comput. Sci. 264 (5) (2011) 23–45.