Proceedings of the 2013 Federated Conference on
Computer Science and Information Systems pp. 377–384

Quadratic TSP: A lower bounding procedure and a column generation approach

Borzou Rostami, Federico Malucelli
Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milan, Italy
Email: rostami@elet.polimi.it, malucell@elet.polimi.it

Pietro Belotti
Department of Mathematical Sciences, Clemson University, Clemson, USA

Stefano Gualandi
Dipartimento di Matematica, Università degli Studi di Pavia, Pavia, Italy

***Abstract*—In this paper we present a Column Generation approach to the Quadratic Traveling Salesman Problem. Given a graph and a function that maps every pair of edges to a cost, the problem consists in finding a cycle that visits every vertex exactly once and such that the sum of the costs over all pairs of consecutive edges of the cycle is minimum. We propose a 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 solve our formulation via column generation. Computational results on some set of instances used in the literature show that our approach is promising. As it obtains a lower bound close to the optimal solutions for all instances.**

I. Introduction

THE TRAVELING SALESMAN PROBLEM (TSP) is one of the most studied optimization problems. Given an undirected graph $G = (V, E)$ with costs $c_e, e \in E$, the problem consists in finding a cycle C that visits each vertex in V exactly once, and such that the sum of the costs c_e of each edge in C is minimum. In its most common form, the TSP has a linear cost function.

In this paper we study a variant of the TSP that has a quadratic cost function, the so-called Quadratic TSP (QTSP). The input of this problem is an undirected (or directed) graph $G = (V, E)$ and a cost function $r : E \times E \to \Re^+$ that maps every pair of edges (or arcs) to a non-negative cost. The QTSP consists in finding a cycle C of minimum cost that visits every vertex of G exactly once. This problem is NP-hard [4]. We distinguish between Asymmetric QTSP and Symmetric QTSP, depending on the fact that the direction of the cycle matters.

The QTSP was introduced with an application to bioinformatics [4] and also has application in robotics and telecommunications. The QTSP can be viewed as a generalization of the Reload Cost TSP (RTSP) introduced in [1]. 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 tackled in [4] with heuristic algorithms based on well-known heuristics for the TSP, with an ad-hoc branch-and-bound solver, and with a branch-and-cut approach based on a linearization of a 0-1 Quadratic Programming formulation of the problem. In [2] and [3], a polyhedral study is used to develop a branch-and-cut algorithm for symmetric and asymmetric QTSP respectively that are the current state-of-the-art for QTSP.

In this paper we present a Linear Programming formulation of the QTSP 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 formulated 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 CG approach.

In Section II we present mathematical formulations and some linearized models for the symmetric QTSP and the asymmetric QTSP. In Section III we present our Linear Programming formulation of the problem. In Section IV we solve the LP model presented in Section III by a column-generation technique and present different formulations of the pricing subproblems for both the symmetric and the asymmetric cases. Computational results are discussed in Section V.

II. The Quadratic Traveling Salesman Problem

In this section we present mathematical formulations for both symmetric QTSP (SQTSP) and asymmetric QTSP (AQTSP). We denote the edge incident with vertices i and j as $\{i, j\}$ in the symmetric case while in the asymmetric case the arc from vertex i to vertex j is denoted as (i, j).

A. The Symmetric QTSP and a Linearized formulation

Consider a complete undirected graph G on a vertex set $V = \{1, 2, \ldots, n\}$ and let $r : E \times E \to \Re^+$ be a cost function that maps a pair of edges to a non-negative cost. The cost of a subgraph in G is equal to the sum of the costs of the pairs of edges incident in the same vertex. The SQTSP seeks a tour (i.e., a cycle passing through each vertex exactly once) of minimum cost. We define the binary variable x_{ij} that is equal to 1 if edge $\{i, j\}$ belongs to the minimum cost tour,

978-1-4673-4471-5/$25.00

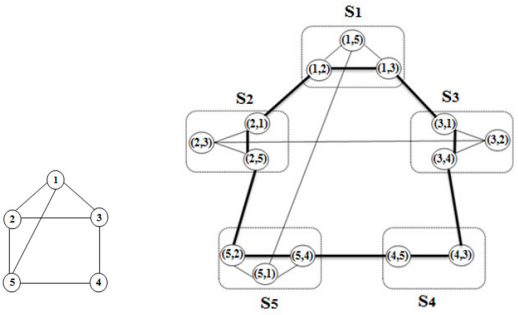

Fig. 1. Graph G and its corresponding *Gadget* graph $\hat{G}$. Note that we assume G to be complete, but this example shows the connections in the gadget graph when G is not complete, or when some of its edges have infinite cost

and 0 otherwise. The SQTSP can be modeled as a integer linear programming (ILP) problem as follows:

$$\min \quad \sum_{\{i,j\}\in E} \sum_{\{j,k\}\in E} r_{ijk} x_{ij} x_{jk} \tag{1}$$

$$\text{s.t.} \quad \sum_{j:\{i,j\}\in E} x_{ij} = 2 \quad \forall i \in V \tag{2}$$

$$\sum_{\substack{\{i,j\}\in E:\\ i\in S, j\in S}} x_{ij} \leq |S| - 1 \quad \emptyset \neq S \subset V \tag{3}$$

$$x_{ij} \in \{0,1\} \quad \forall \{i,j\} \in E. \tag{4}$$

The objective function is straightforward and considers all costs between edges incident on the same vertices. Constraints (2) ensure that each vertex has degree two in the tour, and constraints (3) ensure that no subtour is formed among the subsets of vertices.

A simple linear relaxation of (1)–(4) can be obtained by replacing the term $x_{ij}x_{jk}$ with a binary variable u_{ijk} subject to the following constraints.

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

Motivated by the desire to develop a linearized formulation, we reformulate the SQTSP by creating a new graph $\hat{G} = (\hat{V}, \hat{E})$, called the Gadget graph, as follows:

- For each edge in the graph G, create two nodes $\langle i,j\rangle$ and $\langle j,i\rangle$ and add them to $\hat{V}$.
- For each node $i \in V$ in G, define a super node $S_i = \{\langle i,j\rangle : j = 1,2,\ldots,n, j \neq i\}$. This is only an aggregation of nodes of $\hat{V}$.
- For any $i,j,k \in V$, create an edge between each two nodes $\langle i,j\rangle$ and $\langle i,k\rangle$ in the super node S_i and assign the weight r_{jik} to it.
- For any $i,j \in V$, create an edge between each two nodes $\langle i,j\rangle$ and $\langle j,i\rangle$ in super node S_i and S_j respectively and assign null weight to it.

In Figure 1 we present an example of a graph G and the corresponding *Gadget* graph $\hat{G}$.

By introducing the decision variables u_{jik} to indicate whether edge $\{\langle i,j\rangle, \langle i,k\rangle\}$ in super node S_i is selected or not in the optimal solution, we can rewrite the SQTSP on graph G as the following integer linear problem on the graph $\hat{G}$:

$$\min \quad \sum_{i\in V} \sum_{\substack{j,k\in V:\\ \{\langle i,j\rangle,\langle i,k\rangle\}\in S_i}} r_{jik} u_{jik} \tag{5}$$

$$\text{s.t.} \quad \sum_{\substack{j,k\in V:\\ \{\langle i,j\rangle,\langle i,k\rangle\}\in S_i}} u_{jik} = 1 \quad \forall i \in V \tag{6}$$

$$\sum_{\substack{\langle i,k\rangle\in S_i:\\ k\neq j}} u_{jik} - \sum_{\substack{\langle j,k\rangle\in S_j:\\ k\neq i}} u_{ijk} = 0 \quad \forall \langle i,j\rangle \in \hat{V} \tag{7}$$

$$\sum_{i\in S} \sum_{\substack{\{\langle i,j\rangle,\langle i,k\rangle\}\in \hat{E}:\\ j,k\in S}} u_{jik} \leq |S| - 1 \quad S \subset V \tag{8}$$

$$u_{jik} \in \{0,1\} \quad \forall \{\langle i,j\rangle, \langle i,k\rangle\} \in \hat{E}. \tag{9}$$

Constraints (6) guarantee that within every super node we select exactly one edge, constraints (7) indicate that the number of the edges incident on node $\langle i,j\rangle$ is equal to the number of incident edges on node $\langle j,i\rangle$, and constraints (8) are the subtour elimination constraints. Note that in terms of feasible solutions in the *Gadget graph*, a tour is called *feasible* if it is simple and passes through each super node S_i by visiting exactly one edge of S_i. In the example shown in Figure 1, the bold lines illustrate a feasible tour (a subtour of $\hat{G}$) corresponding to the feasible tour $\{(1,2),(2,5),(5,4),(4,3),(3,1)\}$ in the original graph G.

An alternative model in the case of reload costs spanning tree was studied in [10], where it is assumed that the triangle inequality holds for reload costs at each node of the graph.

B. The Asymmetric QTSP and Linearized formulation

Suppose that the given graph G is a complete directed graph on the vertex set $V = \{1,2,\ldots,n\}$, and let $r : E \times E \to \Re^+$ be a cost function that maps every pair of arcs to a non negative integer cost. The Asymmetric QTSP (AQTSP) can be modeled as follows:

$$\min \quad \sum_{(i,j)\in E} \sum_{(j,k)\in E} r_{ijk} x_{ij} x_{jk} \tag{10}$$

$$\text{s.t.} \quad \sum_{(i,j)\in E} x_{ij} = 1 \quad \forall i \in V \tag{11}$$

$$\sum_{(i,j)\in E} x_{ij} = 1 \quad \forall j \in V \tag{12}$$

$$\sum_{i\in S, j\notin S:(i,j)\in E} x_{ij} \geq 1 \quad S \subset V \tag{13}$$

$$x_{ij} \in \{0,1\} \quad \forall (i,j) \in E, \tag{14}$$

where the binary variable x_{ij} is equal to 1 if the arc (i,j) belongs to the minimum cost tour. Constraints (11) and (12) force to select a single outgoing arc and a single incoming arc for each node, respectively, and constraints (13) are the well known subtour elimination constraints.

In order to linearize the model, we follow the idea of constructing an auxiliary graph as in the symmetric case. We

call this auxiliary graph the *extended graph* and denote it as $\overline{G}$. For each arc (i,j) in the graph G we create a node $\langle i,j\rangle$ in $\overline{G}$, a link between each two nodes $\langle i,j\rangle$ and $\langle j,k\rangle$ in $\overline{G}$ and assign a weight r_{ijk} to each edge $(\langle i,j\rangle,\langle j,k\rangle)$ in $\overline{G}$. If in $\overline{G}$ we partition the set of nodes into n clusters $\mathcal{V}_1,\mathcal{V}_2,\dots\mathcal{V}_n$ such that $\mathcal{V}_i=\{\langle i,j\rangle : j=1,2,\dots,n, j\neq i\}$ for $i=1,2,\dots,n$, then all arcs are defined between nodes $\langle i,j\rangle$ and $\langle j,k\rangle$ from different clusters such that $r_{ijk}>0$; therefore there are no intra-set arcs.

Proposition 2.1: Any feasible tour in G corresponds to a tour in $\overline{G}$ that goes through each cluster exactly once.

Figure 2 represents the extended graph, $\overline{K}_4$, of a directed complete graph, K_4. The bold lines illustrate a feasible tour.

Definition 2.1: Given a directed graph $G=(V,E)$ and a partition $\{\mathcal{V}_i : i=1,\dots,k\}$ of the set V such that $\bigcap_{i=1}^{k}\mathcal{V}_i=\emptyset$ and $\bigcup_{i=1}^{k}\mathcal{V}_i=V$, the *Asymmetric Generalized* TSP (AGTSP) can be stated as the problem of finding a feasible cycle $T\subset E$ which includes exactly one node from each cluster $\mathcal{V}_i$, $i=1,\dots,k$, and whose global cost $\sum_{e\in T}r_e$ is minimum. Therefore the AGTSP involves two related decisions: (i) choosing a node subset $S\subset V$ such that $|S\cap\mathcal{V}_i|=1$ for all $i=1,2,\dots,n$ and (ii) finding a minimum cost Hamiltonian cycle in the subgraph of G induced by S.

Corollary 2.2: Solving the AQTSP on graph G is equivalent to solving the AGTSP on $\overline{G}$.

Using Corollary 2.2, instead of solving the original AQTSP one can solve an AGTSP on graph $\overline{G}$ which is again NP-hard, as it can be reduced to an Asymmetric TSP [8], [9].

We can formulate an integer linear programming model for the AGTSP. Variables u_{ijk} indicate whether arc $(\langle i,j\rangle,\langle j,k\rangle)$ is selected or not in the optimal solution, and variables y_{ij} indicate whether node $\langle i,j\rangle$ is visited or not. The problem is displayed as follows:

GTSP1:

$$\min \sum_{(\langle i,j\rangle,\langle j,k\rangle)\in\overline{E}} r_{ijk}u_{ijk} \tag{15}$$

$$\text{s.t.} \sum_{\substack{j\in V:\\ \langle i,j\rangle\in\mathcal{V}_i}} y_{ij}=1 \quad \forall i\in V \tag{16}$$

$$\sum_{\substack{k\in V:\\ \langle j,k\rangle\in\overline{V}}} u_{ijk}=y_{ij} \quad \forall\langle i,j\rangle\in\overline{V} \tag{17}$$

$$\sum_{\substack{k\in V:\\ \langle k,i\rangle\in\overline{V}}} u_{kij}=y_{ij} \quad \forall\langle i,j\rangle\in\overline{V} \tag{18}$$

$$\sum_{i\in S}\sum_{\substack{j\notin S:\\ \langle i,j\rangle\in\mathcal{V}_i}}\sum_{\substack{k\in V:\\ \langle j,k\rangle\in\mathcal{V}_j}} u_{ijk}\geq 1 \quad S\subset V \tag{19}$$

$$u_{ijk}\in\{0,1\} \quad \forall(\langle i,j\rangle,\langle j,k\rangle)\in\overline{E} \tag{20}$$

$$y_{ij}\in\{0,1\} \quad \forall\langle i,j\rangle\in\overline{V}. \tag{21}$$

Constraint (16) guarantees that from every cluster we select exactly one node. Constraints (17) and (18) require a solution to include exactly one of the arcs entering and exactly one of the arcs leaving the node $\langle i,j\rangle$ if this node is visited. Finally, constraint (19) eliminates all subtours.

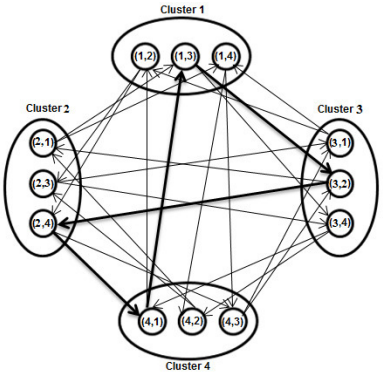

Fig. 2. Extended graph $\overline{K}_4$ of the complete graph K_4

Note that relaxing the integrality requirement on the variables u_{ijk} for all $(\langle i,j\rangle,\langle j,k\rangle)\in\overline{E}$ does not have any effect on problem GTSP1.

By eliminating variables y_{ij} we can write the problem as follows:

GTSP2:

$$\min \sum_{(\langle i,j\rangle,\langle j,k\rangle)\in\overline{E}} r_{ijk}u_{ijk} \tag{22}$$

$$\text{s.t.} \sum_{j\in V}\sum_{\substack{k\in V:\\ (\langle i,j\rangle,\langle j,k\rangle)\in\overline{E}}} u_{ijk}=1 \quad \forall i\in V \tag{23}$$

$$\sum_{k\in V}\sum_{\substack{j\in V:\\ (\langle k,i\rangle,\langle i,j\rangle)\in\overline{E}}} u_{kij}=1 \quad \forall i\in V \tag{24}$$

$$\sum_{\substack{k\in V:\\ \langle j,k\rangle\in\overline{V}}} u_{ijk}-\sum_{\substack{k\in V:\\ \langle k,i\rangle\in\overline{V}}} u_{kij}=0 \quad \forall\langle i,j\rangle\in\overline{V} \tag{25}$$

$$(19),(20). \tag{26}$$

Constraint (23) requires a solution to include exactly one of the arcs entering a cluster, while constraint (24) requires a solution to include exactly one of the arcs leaving a cluster. Constraint (25) is equivalent to network flow conservation constraints and ensure that a solution tour is uninterrupted and continuous.

Theorem 2.3: Constraints (23) in problem GTSP2 are redundant and can be removed from the model.

Proof: Suppose u^* is a feasible solution for problem GTSP2 after removing constraint (23) and define, for each $i\in V$,

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

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

Consider cluster $\mathcal{V}_s$. Then constraint (24) is satisfied for

cluster $\mathcal{V}_s$ by u^*, i.e.,

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

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 (25) 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 2.3 GTSP2 simplifies as follows:

$$\text{GTSP3:}\quad \min \sum_{(\langle i,j\rangle,\langle j,k\rangle)\in\overline{E}} r_{ijk}u_{ijk} \tag{27}$$

$$\text{s.t.}\quad (24),(25),(26). \tag{28}$$

III. Cycle Reformulation of QTSP

In this section we present a new formulation of the QTSP which is based on a cycle generation approach in the given graph. Let C be a cycle of G represented by the set of edges (arcs) that appear in the cycle. The cost of cycle C is $r(C)=\sum_{i,j,k\in V:(i,j),(j,k)\in C} r_{ijk}$, i.e. the sum of the costs of the pairs of consecutive edges (arcs) contained in the cycle. 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 $\min_{C\in\mathcal{C}} c(C)\le\min_{T\in\mathcal{T}} c(T)$.

Following the approach of Held and Karp to the TSP [6], we add a penalty π_i to every vertex i in V, and denote by $\pi(C)=\sum_{i\in C}\pi_i$. Let us consider a new cost function defined as follows: $d(C)=c(C)+\pi(C)$. Let T^* denote an optimal tour of G, i.e. $c(T^*)=\min_{T\in\mathcal{T}} c(T)$. Then, the following relations hold:

$$\min_{C\in\mathcal{C}} d(C)=\min_{C\in\mathcal{C}}\{c(C)+\sum_{i\in C}\pi_i\}\le d(T^*)=c(T^*)+\sum_{i\in V}\pi_i$$

$$\min_{C\in\mathcal{C}}\{c(C)-\sum_{i\in V\setminus C}\pi_i\}\le c(T^*).$$

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

$$\max_{\pi\in\Re^n}\ \min_{C\in\mathcal{C}}\ \{c(C)-\sum_{i\in V\setminus C}\pi_i\}. \tag{29}$$

We describe an LP equivalent to (29) by introducing a variable z as follows:

$$\text{P:}\max\quad z \tag{30}$$

$$\text{s.t.}\quad z+\sum_{i\in V\setminus C}\pi_i\le c(C)\qquad \forall C\in\mathcal{C} \tag{31}$$

$$z,\pi\text{ unrestricted.} \tag{32}$$

Let λ_C be the dual multiplier of constraint (31). The dual of problem P is called master problem and has the following form:

$$\text{D1:}\min\quad \sum_{C\in\mathcal{C}} c(C)\lambda_C \tag{33}$$

$$\text{s.t.}\quad \sum_{C\in\mathcal{C}:i\notin C}\lambda_C=0\qquad \forall i\in V \tag{34}$$

$$\sum_{C\in\mathcal{C}}\lambda_C=1 \tag{35}$$

$$\lambda_C\ge 0\qquad \forall C\in\mathcal{C}. \tag{36}$$

Since all multipliers in (34) are non-negative, and, in each iteration of a column generation approach, exactly one column is generated (as we explain in Section IV), an optimal solution to the problem 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 $c(C^*)$ provides a lower bound for the original QTSP.

By subtracting each constraint (34) from (35) and removing (35) from D1, one can find a relaxation of the problem D1 as follows:

$$\text{D2:}\min\quad \sum_{C\in\mathcal{C}} c(C)\lambda_C \tag{37}$$

$$\text{s.t.}\quad \sum_{C\in\mathcal{C}:i\in C}\lambda_C=1\qquad \forall i\in V \tag{38}$$

$$\lambda_C\ge 0\qquad \forall C\in\mathcal{C}. \tag{39}$$

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

IV. Column Generation Approach

In this section we develop a column generation approach to solve problems D1 and D2. Since the number of cycles in $\mathcal{C}$ is exponential with respect to the number of vertices, we first consider a restricted version of the master problem (RMP) 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 (38). 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 (34) and (35) respectively. The reduced cost of the variable λ_C for each $C\in\mathcal{C}$ is $\overline{r(C)}=r(C)-(\pi(C))'-z$, where $(\pi(C))'=\sum_{i\notin C}\pi_i$. A column to enter the basis can be found by computing a minimum cost cycle with respect to $r_{ijk}+\pi_j$ for each $(i,j)\in E,(j,k)\in E$. Let $\overline{r(C^p)}=\min_{C\in\mathcal{C}}\overline{r(C)}$. If $\overline{r(C^p)}\ge 0$ then the current solution is optimal. Otherwise we select column C^p to enter the basis.

Theorem 4.1: If the column C^p to enter 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

$$\overline{r(C^p)}=r(C^p)-(\pi(C^p))'-z\le r(C)-(\pi(C))'-z. \tag{40}$$

If cycle C is also a tour, then $r(C^p)\le r(C)$. ■

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

$$\overline{r(C)}=r(C)-\pi(C). \tag{41}$$

A. Pricing subproblems

A column to enter the basis can be found by computing a minimum cost cycle in the original graph G with respect to $r_{ijk} + \pi_j$. 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 [5]. In this section we explain how to update the linearized models, presented in Section II, to solve the pricing problems. In order to define a suitable model for the pricing subproblem of the restricted master problem D1, we consider the SQTSP and AQTSP separately.

a) Symmetric case: Consider the pricing problem of D1 in the symmetric case. As we mention in Section II-A, instead of looking for a cycle in the original graph one can easily find a cycle in the *Gadget* graph; i.e., finding a negative reduced cost of problem D1 is the same as finding a negative cost feasible cycle in the Gadget graph. Defining a binary variable w_i to indicate whether the super node S_i is on the cycle or not, the minimum negative cost cycle is then found by solving the following problem:

$$\min \quad \sum_{i\in V} \sum_{\{\langle i,j\rangle,\langle i,k\rangle\}\in S_i} r_{jik}u_{jik} - \sum_{i\in V}\pi_i(1-w_i) - z$$

$$\text{s.t.} \quad \sum_{\substack{j,k\in V:\\ \{\langle i,j\rangle,\langle i,k\rangle\}\in S_i}} u_{jik} = w_i \quad \forall i \in V \tag{42}$$

$$(7)-(9) \tag{43}$$

$$w_i \in \{0,1\} \quad \forall i \in V. \tag{44}$$

Considering the definition of the w variables, one can obtain an equivalent formulation for the pricing problem by replacing constraint (42) with the so-called resource constraint $\sum_{\substack{j,k\in V\\ \{\langle i,j\rangle,\langle i,k\rangle\}\in S_i}} u_{jik} \le 1$ and removing the variables w from the model. Also, it should be observed that by forcing the solution of the pricing problem to be a cycle with negative cost one can easily remove the constraint (8) from the pricing model. Therefore, finding a cycle with negative reduced cost is simply a matter of finding a path between each node of the *Gadget* graph and itself with a negative cost which satisfies the resource constraint. Since the dual variables π are defined on each super node of the *Gadget* graph, the problem of finding a negative reduced cost cycle can be formulated as resource-constrained elementary shortest path problem. Let q_{st} denote the cost of the resource-constrained elementary shortest path from origin node $\langle s,t\rangle$ to itself in the *Gadget* graph. The pricing problem can be written as: $\min_{\langle s,t\rangle\in\hat{V}}\{q_{st}\}$, where q_{st} is the optimal value of the following problem.

$$\min \quad \sum_{i\in V} \sum_{\{\langle i,j\rangle,\langle i,k\rangle\}\in S_i} (r_{jik}+\pi_i)u_{jik} - \sum_{i\in V}\pi_i - z \tag{45}$$

$$\text{s.t.} \quad \sum_{\substack{j\in V:\\ \{\langle s,t\rangle,\langle s,j\rangle\}\in S_i}} u_{tsj} = 1 \tag{46}$$

$$\sum_{\substack{j\in V:\\ \{\langle s,j\rangle,\langle s,t\rangle\}\in S_i}} u_{jst} = 1 \tag{47}$$

$$\sum_{\substack{\langle i,k\rangle\in S_i:\\ k\neq j}} u_{jik} - \sum_{\substack{\langle j,k\rangle\in S_j:\\ k\neq i}} u_{ijk} = 0 \quad \forall \langle i,j\rangle \neq \langle s,t\rangle \tag{48}$$

$$\sum_{\substack{j,k\in V\\ \{\langle i,j\rangle,\langle i,k\rangle\}\in S_i}} u_{jik} \le 1 \quad \forall i \in V \tag{49}$$

$$u_{jik} \in \{0,1\} \quad \forall\{\langle i,j\rangle,\langle i,k\rangle\} \in \hat{E}. \tag{50}$$

Constraints (46), (47), (48) and (50) find a path from the source node $\langle s,t\rangle$ to itself. The resource constraint (49) guarantee that each super node S_i is visited at most once.

b) Asymmetric case: In the asymmetric case, finding a negative reduced cost directed cycle for problem D1 is equivalent to solving the modified version of the GTSP1 or GTSP3 explained in Section II-B. Here we provide a version of the latter so that the resulting problem gives the most negative directed cycle.

$$\min \quad \sum_{(\langle i,j\rangle,\langle j,k\rangle)\in\overline{E}} r_{ijk}u_{ijk} - \sum_{i\in V}\pi_i(1-t_i) - z$$

$$\text{s.t.} \quad \sum_{k\in V} \sum_{\substack{j\in V\\ (\langle k,i\rangle,\langle i,j\rangle)\in\overline{E}}} u_{kij} = t_i \quad \forall i \in V$$

$$(20),(25),(24)$$

$$t_i \in \{0,1\} \quad \forall i \in V.$$

The binary variable t_i is equal to 1 if cluster i is visited, otherwise it is zero. Following the same process as the symmetric case, one can obtain an equivalent formulation based on resource-constrained elementary shortest path problem in the extended graph.

The solution of the subproblems provides either a certificate of optimality of the current solutions (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 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 [16]. 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.

B. Stabilized column generation

Column generation methods usually suffer from slow convergence to the optimal solution. Primal degeneracy, dual degeneracy and instability in the behavior of dual variables are well known to cause slow convergence and *tailing off* effects to column generation procedures [11], [12].

To control the dual variables during the solution process, we use the stabilized column generation approach proposed in [15]. This approach combines the box step method [14] with a

kind of descent method proposed in [13]. 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 $\bar{\mathcal{C}} \in \mathcal{C}$, as the following model:

$$\text{RD1: min} \quad \sum_{C \in \bar{\mathcal{C}}} c(C)\lambda_C \tag{51}$$

$$\text{s.t.} \quad \sum_{C \in \bar{\mathcal{C}}: i \in C} \lambda_C \geq 1 \qquad \forall i \in V \tag{52}$$

$$(35), (36).$$

Note that since the set partitioning constraints allow 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 (52) 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:

$$\text{SP: max} \quad z + \sum_{i \in V} \pi_i - \varepsilon_i^- w_i^- - \varepsilon_i^+ w_i^+ \tag{53}$$

$$\text{s.t.} \quad z + \sum_{i \in C} \pi_i \leq c(C) \qquad \forall C \in \bar{\mathcal{C}} \tag{54}$$

$$\pi_i + w_i^- \geq \delta_i^- \qquad \forall i \in V \tag{55}$$

$$\pi_i - w_i^+ \leq \delta_i^+ \qquad \forall i \in V \tag{56}$$

$$\pi, w^-, w^+ \geq 0, \ z \text{ unrestricted.} \tag{57}$$

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

$$\text{SD1: min} \quad \sum_{C \in \bar{\mathcal{C}}} c(C)\lambda_C + \sum_{i \in V} -\delta_i^- \mu_i^- + \delta_i^+ \mu_i^+ \tag{58}$$

$$\text{s.t.} \quad \sum_{C \in \bar{\mathcal{C}}: i \in C} \lambda_C - \mu_i^- + \mu_i^+ \geq 1 \quad \forall i \in V \tag{59}$$

$$\sum_{C \in \bar{\mathcal{C}}} \lambda_C = 1 \tag{60}$$

$$\mu_i^- \leq \varepsilon_i^- \quad \forall i \in V \tag{61}$$

$$\mu_i^+ \leq \varepsilon_i^+ \quad \forall i \in V \tag{62}$$

$$\lambda, \mu^-, \mu^+ \geq 0. \tag{63}$$

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 (60) and is called SD2. In order to use the stabilized models efficiently, one must initialize and update the parameters correctly. With desire 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 to the problem, 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 V we discuss more about the updating process.

V. Computational Experiments

In this section we present our computational experiments on a class of randomly generated instances with size rating from $n = 5$ to $n = 25$ introduced in [4]. All instances consist of complete graphs with n vertices and $m = n(n-1)/2$ edges. The cost function $r : E \times E \to \Re_0^+$ for both symmetric and asymmetric instances is defined as an integer number which is chosen from the set $\{0, 1, ..., 10000\}$ uniformly for all $(i, j), (j, k) \in E$ such that $i \neq k$ and set to infinity for all $(i, j), (j, i) \in E$. We used the AMPL modeling language [17] with GUROBI 5.0.0 [18] 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.

A. Stabilization

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 $(\tilde{\pi} - 100, \tilde{\pi} + 100)$, ($\tilde{\pi}$ 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 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 criteria of the stabilized column generation algorithm is the same as case of problem SD1.

TABLE I
COMPUTATIONAL TIME OF COLUMN GENERATION (CG) AND STABILIZED CG APPROACHES FOR BOTH THE SQTSP THE AQTSP INSTANCES

	CPU time (Symmetric)				CPU time (Asymmetric)			
size	CG2	SCG2	CG1	SCG1	CG2	SCG2	CG1	SCG1
10	2.15	1.91	4.36	4.12	3.91	3.26	8.93	7.35
20	43.57	33.34	205.94	60.70	97.43	34.92	508.53	271.81

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 I. 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.

B. Lower bound Computation

As we mentioned in Section III, a solution of the pricing problem, which may contain a single or multiple cycles, 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. In other words, we allow subtours in the optimal solution of the original QTSP, which is in fact a relaxation of the problem. Therefore, when no more new columns can be priced out, a solution of the master problem RD1 gives a lower bound on the original problem. Note that the optimal value of the problem RD2 always gives a lower bound on the original problem, regardless of the solution being a single cycle or multiple ones.

In Table II we present computational results of the lower bounding schemes for both the symmetric and the asymmetric QTSPs. Each row of the tables reports the average results over ten instances of the same size. The problem size 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. We compare three different average lower bounds (LB) on the optimal objective values, their computing time (time), number of iterations (iter), and the average gap. The first lower bound LB(LP) in column three is the lower bound obtained with the linear relaxation of problem (5) – (9) and the linear relaxation of problem GTSP3 for the SQTSP and the AQTSP respectively. The computation time of the LP relaxation is less than two seconds for all instances; therefore we did not mention it in the table. The second lower bound is obtained via the Stabilized Column Generation approach to the problem SD2 (SCG2); and the third lower bound is obtained via the Stabilized Column Generation approach apply to the problem SD1 (SCG1). Columns four to seven and columns eight to eleven represent the optimal value, computation time, number of iterations needed to identify the optimal solution and the gap. The formula we used to compute these gaps is $(opt - LB(SCG))/(opt - LB(LP))$, where opt and $LB()$ stand for the optimal value and the lower bound, respectively. We can see in this table that the bounds obtained by SCG1 outperforms the other two in all instances and are close to the optimal values in both the SQTSP and the AQTSP. Also the lower bond obtained by SCG2 is indeed better than the one obtained with LP relaxation except for the instances of dimension five, for which the LP relaxation gives on average a tighter bound. On average the ratio of gap between the lower bound obtained by SCG2 and the optimal solution, and the gap between the lower bound obtained by linear relaxation and the optimal solution for the symmetric instances is 0.72, while this ratio for the asymmetric instances is 0.73. As we see, on average, the improvement of lower bound by applying the SCG2 for both symmetric and the asymmetric cases is almost the same, while the computational time for the asymmetric instances is more than the computational time for the symmetric ones.

According to Table II, applying SCG1 yields a considerable improvement of the lower bounds in both the symmetric and the asymmetric cases, i.e., on average the ratio of gap between the lower bound obtained by SCG1 and the optimal solution over the gap between the lower bound obtained by linear relaxation and the optimal solution for the symmetric instances is 0.09, and for the asymmetric instances is 0.20. These gaps show the improvement of lower bounds in both the symmetric and asymmetric cases in compare to the SCG2. Also it should be noted that the improvement of the lower bounds and also the computational time in the symmetric case is more attractive than in the asymmetric case.

VI. CONCLUSIONS

In this paper we first proposed two different linearization models to the SQTSP and the AQTSP. We also presented a different cycle formulation for the QTSP (in general) and solved the resulting LP problem by Column Generation approach. We have shown how the linearized formulations can be applied to finding the negative reduced cost in the pricing problem. To overcome the problems of instability in the behavior of dual variables of the presented master problem, we used a stabilized column generation approach. Our experiments show that our column generation approach outperforms the LP relaxation of the QTSP in both the symmetric and the asymmetric cases. The main goal of this paper is to show the advantage of column generation and the weakness of the LP relaxation in finding a good lower bound for the QTSP.

TABLE II
COMPARISON OF THREE DIFFERENT LOWER BONDING APPROACHES

			$SCG2$				$SCG1$			
$size$	$Opt.$	$LB(LP)$	LB	$time$	$iter$	Gap	LB	$time$	$iter$	Gap
Symmetric:										
5	14922.2	13839.5	13606.3	0.62	6	1.21	14922.2	3.89	65	0.0
6	12217.5	11817.1	11816.2	0.53	7	0.99	12160.4	3.35	43	0.14
7	14648.6	13200.4	13747.9	0.85	11	0.62	14356.5	3.51	41	0.20
8	15055.7	12544.1	12623.9	1.12	14	0.96	14542.6	4.06	43	0.20
9	14562.2	11909.4	12710.7	1.91	16	0.73	14225.7	4.12	37	0.12
10	15018.6	11939.9	13104.5	1.77	18	0.62	14718.7	4.86	45	0.09
11	13819.6	10367.8	11175.9	1.75	17	0.76	12958.9	6.37	33	0.24
12	14719.4	10896.2	12036.6	3.15	21	0.70	13740.2	7.96	31	0.25
13	13174.3	9826.6	10954.4	4.90	23	0.66	12705.5	12.11	32	0.14
14	13548.7	9823.7	11089.7	9.34	25	0.66	12974.4	13.83	32	0.15
15	12531.0	9354.7	10697.8	12.38	28	0.57	12219.3	14.20	25	0.09
16	13426.1	9065.7	10253.3	13.24	26	0.72	12573.4	20.56	28	0.19
17	13022.5	9324.8	10420.6	18.55	30	0.70	12735.9	27.35	32	0.07
18	12388.6	8522.8	9831.7	22.35	34	0.66	12137.6	28.38	29	0.06
19	12697.5	8630.6	10036.8	29.98	39	0.65	12394.9	45.92	34	0.07
20	13246.0	8797.3	10092.8	33.34	40	0.70	12491.8	60.70	35	0.16
21	12699.4	8352.0	9868.1	45.71	45	0.65	12260.4	75.16	34	0.10
22	12032.2	8078.2	9519.4	50.81	47	0.63	11859.1	98.45	32	0.04
23	12378.4	8123.9	9376.2	53.98	46	0.70	11911.7	154.01	37	0.10
24	11871.1	7996.5	9250.5	65.78	51	0.67	11480.9	203.40	41	0.10
25	11673.5	7494.2	8717.8	81.18	54	0.70	11187.1	172.77	28	0.11
Asymmetric:										
5	12372.2	10758.2	11126.8	1.80	46	0.77	12373.2	2.10	56	0.0
6	11883.1	10291.6	10596.9	1.79	38	0.80	10684.6	1.51	37	0.75
7	13204.9	10486.3	11369.9	1.95	42	0.67	12746.3	2.98	55	0.16
8	13363.4	10116.3	11194.4	2.08	38	0.66	12878.5	3.24	45	0.14
9	13063.2	9610.1	10546.5	2.52	37	0.72	12135.1	5.24	48	0.26
10	12921.5	9427.1	10461.9	3.26	33	0.70	12500.8	7.35	43	0.12
11	12997.5	8965.8	9891.20	4.41	34	0.77	12089.9	11.99	49	0.22
12	11434.8	8723.3	9682.4	5.79	29	0.64	10897.9	10.87	35	0.19
13	12171.5	8421.6	9608.1	8.55	33	0.68	11584.3	21.11	41	0.15
14	11838.3	8182.7	9005.6	9.95	37	0.77	10635.6	20.66	38	0.32
15	12428.8	8094.6	9564.7	16.83	35	0.66	11748.8	37.05	45	0.15
16	12135.7	7726.8	8764.6	16.64	35	0.76	11119.8	47.34	51	0.23
17	11832.2	7241.3	8682.7	21.69	35	0.68	10094.7	60.02	50	0.37
18	11662.2	7380.9	8544.8	24.40	36	0.72	11104.6	119.24	59	0.13
19	12095.9	7264.6	8560.4	31.22	38	0.73	11207.8	157.51	57	0.18
20	11802.8	6951.1	8200.1	34.92	37	0.74	11162.4	271.81	63	0.13
21	11288.2	6948.5	8140.5	51.44	40	0.72	10819.1	435.246	58	0.10
22	11741.0	6906.8	7950.1	51.61	43	0.78	10517.2	480.15	58	0.25
23	11549.7	6821.3	7688.3	59.05	45	0.81	10549.3	671.15	68	0.21
24	11239.1	6580.4	7716.1	84.03	43	0.75	10180.1	877.83	69	0.22
25	11434.8	6663.1	7785.6	105.36	44	0.76	10422.3	1698.31	70	0.21

REFERENCES

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

[2] Fischer, A., and C. Helmberg, "The symmetric quadratic traveling salesman problem," F. Mathematik, T.U. Chemnitz, Preprint 2011-8, 2011.

[3] Fischer, A., "The asymmetric quadratic traveling salesman problem," F. Mathematik, T.U. Chemnitz, Preprint 2011-19, 2011.

[4] Fischer, F., G. Jäger, A. Lau, and P. Molitor, "Complexity and algorithms for the traveling salesman problem and the assignment problem of second order," F. Mathematik, T.U. Chemnitz, Preprint 2009-16, 2009.

[5] Galbiati, G., S. Gualandi, and F. Maffioli, *On minimum reload cost cycle cover*, Electronic Notes in Discrete Mathematics, *36* (2010), 81–88.

[6] Held, M., and R. M. Karp, *The Traveling-Salesman Problem and Minimum Spanning Trees*, Operations Research, *18* (1970), 1138–1162.

[7] Jäger, G., and P. Molitor, *Algorithms and experimental study for the traveling salesman problem of second order*, Lecture Notes in Computer Science, *5165* (2008), 211–224.

[8] Laporte, G., H. Mercure, Y. Nobert, *Generalized Traveling Salesman Problem through n set of nodes: the assymmetric case*, Discrete Applied Mathematics, *18* (1987), 185–197.

[9] Noon, C.E, and J.C. Bean, *A Lagrangian Based approach for the Assymmetric Generalized Traveling Salesman Problem*, Operations Research, *39* (1991), 623–632.

[10] Gamvros, I., L. Gouveia, and S. Raghavan, *Reload Cost Trees and Network Design*, Networks, *59* (2012), 365–379.

[11] Gilmore, P.C, and R.E. Gomory, *A Linear Programming approach to the Cutting-stock problem*, Operations Research, *9* (1961), 849–859.

[12] Kallehauge, B., J. Larsen, O. B. G. Madsen,"Lagrangian duality applied on wehicle routing with time windows", Technical report IMMM-IR-2001-9, Information and Mathematical Modeling, Technical University of Denmark, KGS, Lyngby, Denmark, 2001.

[13] Kim, S., K.-N. Chang, J.-Y. Lee, *A descent method with linear programming subproblems for nondifferentiable convex optimization*, Math. Programming, *71* (1995), 17–28.

[14] Marsten, R. E., W.W. Hogan, J.W. Blankenship, *The BOXSTEP method for large-scale optimization*, Operations Research, *9* (1975), 389–405.

[15] Du Merle, O., D. Villeneuve, J. Desrosiers, P. Hansen, *Stabilized column generation, Discrete Math. 194* (1999), 229–237.

[16] Barnhart, C., Johnson, E.L., Nemhauser, G.L., Savelsbergh, M.W.P. and Vance, P.H., *Branch-and-Price: Column Generation for Solving Huge Integer Programs*, Operations Research *46* (1998), 316–329.

[17] Fourer, R., D. M. Gay, B. W. Kernighan, *AMPL: A Modeling Language for Mathematical Programming*, Duxbury Press, ISBN 978-0-534-38809-6, (2002).

[18] http://www.gurobi.com/documentation/5.5/reference-manual.