Computers & Operations Research 37 (2010) 1762–1773

Contents lists available at ScienceDirect

Computers & Operations Research

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

The quadratic minimum spanning tree problem: A lower bounding procedure and an efficient search algorithm

Temel Öncan [a,*], Abraham P. Punnen [b]

[a] Endüstri Mühendisligi Bölümü, Galatasaray Üniversitesi, Ortaköy, İstanbul 34357, Turkey
[b] Department of Mathematics, Simon Fraser University, 13450 102nd AV, Surrey, British Columbia, Canada V3T 5X3

ARTICLE INFO

Available online 18 January 2010

Keywords:
Quadratic minimum spanning tree problem
Lagrangian relaxation
Local search

ABSTRACT

In this paper we consider the quadratic minimum spanning tree problem (QMSTP) which is known to be NP-hard. Given a complete graph, the QMSTP consists of finding a minimum spanning tree (MST) where interaction costs between pairs of edges are prescribed. A Lagrangian relaxation procedure is devised and an efficient local search algorithm with tabu thresholding is developed. Computational experiments are reported on standard test instances, randomly generated test instances and quadratic assignment problem (QAP) instances from the QAPLIB by using a transformation scheme. The local search heuristic yields very good performance and the Lagrangian relaxation procedure gives the tightest lower bounds for all instances when compared to previous lower bounding approaches.

1. Introduction

Consider an undirected complete graph $G=(V,E)$ where $V=\{v_1,\ldots,v_n\}$ is the vertex set and $E=\{e_1,\ldots,e_m\}$ is the edge set. The quadratic minimum spanning tree problem (QMSTP) consists of determining a minimum spanning tree (MST) where interaction costs between pairs of edges are prescribed. The QMSTP has applications in transportation, telecommunication, irrigation and energy distribution. The problem was first introduced by Assad and Xu [20,1] who have shown that it is NP-hard. In their seminal paper, a variant of the QMSTP, namely adjacent only QMSTP (AQMSTP), is also discussed. In that particular case, the interaction costs of all non-adjacent edge pairs are assumed to be zero.

Assad and Xu [1] have devised a lower bounding algorithm and two heuristic algorithms for this problem. They incorporated these algorithms into a branch-and-bound scheme for obtaining exact solutions. The authors have reported that they have solved QMSTP instances up to 15 vertices to optimality. Zhou and Gen [21] have devised a genetic algorithm (GA) using the Prüfer number encoding. They have noted that the best solution obtained by the heuristics devised by Assad and Xu [1] can be improved on average 9.6% and at most 12.83% with their GA approach. The results reported by Zhou and Gen [21] are based on a test bed consisting of 16 randomly generated test instances ranging from n=6 to 50 vertices. Gao and Lu [6] have addressed the fuzzy QMSTP (FQMSTP) and they formulated the FQMSTP as an expected value model with chance-constrained programming and dependent-chance constrained programming. The authors have also proposed a simulation based genetic algorithm using Prüfer number representation for the solution of the FQMSTP. Recently, Soak et al. [18,19] devised novel evolutionary algorithms (EA) using various encoding strategies. Soak et al. [18] reported that for the QMSTP case the EA using adaptive learning technique performed best over several encoding strategies including the Prüfer encoding. The QMSTP has little attracted the interest of the operations research community and hence to the best of our knowledge, there is no other paper published on this problem.

The remainder of this paper is organized as follows. Section 2 introduces two mathematical programming formulations of the QMSTP. Section 3 presents lower bounding procedures for the problem. This is followed in Section 4 by a local search algorithm for the QMSTP. Section 5 is where we present our computational experiments. Finally, we conclude with Section 6.

2. The quadratic minimum spanning tree problem

In this section we present two formulations of the QMSTP proposed by Assad and Xu [1]. The binary variable x_p denotes the presence of the edge e_p in the solution. The first formulation by Assad and Xu [1] is

$$\text{AX1}: \quad \min\left\{ z(x)=\sum_{p=1}^{m}\sum_{\substack{q=1\\ p\neq q}}^{m} a_{pq}x_p x_q+\sum_{p=1}^{m} b_p x_p \;\middle|\; \mathbf{x}\in\mathcal{T}\right\} \tag{1}$$

* Corresponding author.
E-mail addresses: ytoncan@gsu.edu.tr (T. Öncan), apunnen@sfu.ca (A.P. Punnen).

0305-0548/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2010.01.004

where $\mathcal{T}$ represents the set of spanning trees and each spanning tree is represented by an incidence vector $\mathbf{x}$ of size m, i.e. $\mathbf{x}=(x_p)_{p=1,\ldots,m}$. Here, b_p indicates the cost of selecting edge e_p and a_{pq} stands for the interaction cost associated with each pair of selected edges e_p and e_q. Note that if the interaction cost of edges e_p and e_q is a then it is assumed that $a_{pq}=a_{qp}=a/2$ holds for $p,q=1,\ldots,m$ and $p\neq q$. The interaction costs a_{qp} are taken into account in the design of several networks when the interference costs between edges are important.

For their second QMSTP formulation, Assad and Xu [1] have defined additional decision variables y_{pq} equal to 1 if and only if both edges e_p and e_q are in the solution. The formulation is the following:

$$\text{AX2}: \quad \min \sum_{p=1}^{m}\sum_{\substack{q=1\\q\neq p}}^{m} a_{pq}y_{pq}+\sum_{p=1}^{m} b_p x_p \tag{2}$$

$$\text{s.t.} \quad \sum_{q=1}^{m} y_{pq}=(n-1)x_p, \quad p=1,\ldots,m \tag{3}$$

$$\sum_{p=1}^{m} y_{pq}=(n-1)x_q, \quad q=1,\ldots,m \tag{4}$$

$$y_{pp}=x_p, \qquad p=1,\ldots,m \tag{5}$$

$$0\le y_{pq}\le 1, \qquad p,q=1,\ldots,m \tag{6}$$

$$\mathbf{x}\in\mathcal{T} \tag{7}$$

One can observe that, in the objective function (2) the quadratic terms have been eliminated at the cost of the additional decision variables y_{pq}. In this formulation, constraints (3) (constraints (4)) ensure that whenever edge e_p (edge e_q) is selected then the sum of y_{pq} variables over q (over p), and hence the total number of selected edges, must be exactly equal to $(n-1)$. Observe that with constraints (3)–(6) the quadratic terms are eliminated from the objective function.

When $a_{pq}=\mu_p v_q$ and $b_q=0$ for all p and q, the resulting QMSTP is called the multiplicative minimum spanning tree problem (MMSTP). For arbitrary μ_p and v_q, this problem is known to be NP-hard [16]. In [16] a polynomial algorithm is given for special case of the MMSTP where μ_p and $v_p\ge 0$ for all $p=1,\ldots,m$ based on more general algorithms given in [11,17]. The description in [16], however, has an error, although the general algorithm in [17] is correct. For completeness and clarity, we present the corrected algorithm for the MMSTP from [16] here. Consider the parametric problem

$$PT: \quad g(\lambda)=\min_{\mathbf{t}\in\mathcal{T}}\sum_{e\in\mathbf{t}}(\mu_e+\lambda v_e)$$

where $\mathcal{T}$ is the family of all spanning trees of G. It is well known that $g(\lambda)$ is a piece-wise linear convex function. Let $\lambda_1,\lambda_2,\ldots,\lambda_k$ be the break-points of $g(\lambda)$. Let $\mathbf{t}(\lambda_i)$ be the optimal solution to PT when $\lambda=\lambda_i$. Choose r such that

$$\left(\sum_{e\in\mathbf{t}(\lambda_r)}\mu_e\right)\left(\sum_{e\in\mathbf{t}(\lambda_r)}v_e\right)=\min_{1\le i\le k}\left(\sum_{e\in\mathbf{t}(\lambda_i)}\mu_e\right)\left(\sum_{e\in\mathbf{t}(\lambda_i)}v_e\right)$$

Then $\mathbf{t}(\lambda_r)$ is an optimal solution to the MMSTP. Since the number of break-points of $g(\lambda)$ is polynomial, the MMSTP can be solved in polynomial time.

3. Lower bounding procedures

To compute lower bound values for the QMSTP we propose a Lagrangian relaxation scheme based on an extended formulation of the QMSTP. Before introducing our Lagrangian scheme we will present the Gilmore and Lawler type lower bound and the leveling procedure devised by Assad and Xu [1].

3.1. Gilmore and Lawler type lower bound for the QMSTP

One of the earliest lower bounding schemes for the quadratic assignment problem (QAP) is the one proposed by Gilmore [7] and Lawler [14]. Their approach can also be adopted to the QMSTP. For that purpose, we first solve m minimum spanning tree problems (MSTP) by setting $s_q=1$ for each edge e_q. Here, the decision variables s_q equal to 1 if and only if edge e_q is in the solution. In other words, we solve the following m problems, for each edge p with $p=1,\ldots,m$:

$$d_p=\min\sum_{q=1}^{m} a_{pq}s_q \tag{8}$$

$$\text{subject to} \quad \mathbf{s}\in\mathcal{T} \tag{9}$$

$$s_p=1 \tag{10}$$

where $\mathbf{s}$ is the solution vector consisting of s_p values. Then using the solution values d_p of the problems (8)–(10) we solve the following MSTP:

$$GL=\min\sum_{p=1}^{m}(d_p+b_p)x_p \tag{11}$$

$$\text{subject to} \quad \mathbf{x}\in\mathcal{T} \tag{12}$$

Hence, we obtain Gilmore and Lawler type lower bound for the QMSTP. For the QAP case, Gilmore and Lawler bounds are easy to compute but they are weaker than other state-of-the-art lower bounding schemes. Therefore, we do not expect superior results from the Gilmore and Lawler approach for the QMSTP.

3.2. The Assad and Xu's leveling procedure

By using the transformation

$$b_p(\gamma)=b_p-(n-2)\gamma_p, \quad p=1,\ldots,m$$

$$a_{pq}(\gamma)=a_{pq}+\gamma_q, \quad p,q=1,\ldots,m$$

Assad and Xu [1] have proposed the following lower bound for the QMSTP:

$$AX(\gamma)=\min\left\{\sum_{p=1}^{m} f_p(\gamma)x_p \,\middle|\, x\in\mathcal{T}\right\} \tag{13}$$

where

$$f_p(\gamma)=\min\left\{b_p(\gamma)+\sum_{\substack{q=1\\q\neq p}}^{m} a_{pq}(\gamma)x_q \,\middle|\, x_p=1 \text{ and } x\in\mathcal{T}\right\}, \quad p=1,\ldots,m \tag{14}$$

and $\boldsymbol{\gamma}$ is a m dimensional parameter vector. Observe that the lower bound $AX(\gamma)$ proposed by Assad and Xu [1] depends on the parameters γ. To determine the values of the γ parameters yielding the largest lower bound $AX^*=\max_\gamma\{AX(\gamma)\}$, the authors proposed the following algorithm.

The Leveling Procedure (Assad and Xu [1]).

Step 0: A tolerance limit $\varepsilon>0$ is set for the stopping rule and iteration counter i is set to 1. Initially, $\boldsymbol{\gamma}^1=(0,\ldots,0)$.

Step 1: Compute $f_p(\gamma^i)$ for $p=1,\ldots,m$ using (13) and then compute $AX(\gamma^i)$ using (14).

Step 2: If $\max_i\{f_p(\gamma^i)\}-\min_i\{f_p(\gamma^i)\}<\varepsilon$ then STOP, else update γ by setting
$\gamma_p^{i+1}=\gamma_p^i+\frac{1}{n-1}f_p(\gamma^i), \quad p=1,\ldots,m$
Set $i:=i+1$ and go to **Step 1**

The leveling procedure stops when $AX^*-AX(\gamma^i)<\varepsilon$. Observe that when a straightforward implementation of Kruskal algorithm [12] is employed in Step 1, the worst case complexity of this step becomes $\mathcal{O}(m^2\log n)$ because $m+1$ MSTPs must be solved. The complexity of this step reduces to $\mathcal{O}(m(m+n\log n))$ with more sophisticated data structures.

For $\gamma^1=(0,\ldots,0)$, the lower bound value $AX(\gamma^1)$ obtained by the leveling procedure is exactly the same as the lower bound yielded by the Gilmore and Lawler approach. Hence, we can say that the leveling procedure starts with the Gilmore and Lawler bound and iteratively improves it until the stopping criterion is satisfied.

3.3. A Lagrangian relaxation scheme

In this section we propose a Lagrangian relaxation scheme based on an extended formulation with valid inequalities (EFVI). In fact, the proposed formulation is an extension of the AX2 with the following valid inequalities (15) and (16)

$$\sum_{p\in\delta(i)}^{m} y_{pq}\geq x_q, \quad i=1,\ldots,n, \;\; q=1,\ldots,m \tag{15}$$

$$\sum_{q\in\delta(i)}^{m} y_{pq}\geq x_p, \quad i=1,\ldots,n, \;\; p=1,\ldots,m \tag{16}$$

where $\delta(i)$ denotes the set of edges incident to vertex i. In the following proposition, we will show that the constraints (15) and (16) are valid.

Proposition 1. *The constraints* (15) and (16) *are valid for the QMSTP.*

Proof. To show the validity of constraints (15) consider the case when x_q equals to 1 namely when edge q is in the solution. Then at least one of the y_{pq} variables should be equal to 1 because there must be at least one edge p which is incident to a vertex i since the solution $\mathbf{x}$ corresponds to a spanning tree. For the other case, namely when x_q equals to 0 then constraints (15) hold by the definition. The validity of constraints (16) follows similarly. □

For the sake of clarity we should state that the EFVI consists of the objective function (2) subject to constraints (3)–(7), (15) and (16). Note that in both AX2 and EFVI, it is inherent that the relation $y_{pq}=x_px_q$ must hold at optimality (Assad and Xu [1]).

Now consider a Lagrangian relaxation of the EFVI with multipliers α_{iq} for constraints (15). The Lagrangian function $L(\alpha)$ is thus defined by

$$\min \quad \sum_{p=1}^{m}\sum_{\substack{q=1\\q\neq p}}^{m}\overline{c}_{pq}y_{pq}+\sum_{p=1}^{m}\overline{b}_p x_p \tag{17}$$

$$\text{s.t.} \quad (16),\ (3)-(7) \tag{18}$$

where

$$\overline{c}_{pq}=a_{pq}-\alpha_{iq}, \qquad p,q=1,\ldots,m, \;\; i=1,\ldots,n;$$

$$\overline{b}_p=b_p+\sum_{k=1}^{n}\alpha_{kp}, \quad p=1,\ldots,m$$

For the QAP, which has a similar structure to the QMSTP, Frieze and Yadegar [5] have employed a decomposition approach within a Lagrangian relaxation scheme. Inspired with this idea, we adopt their approach as follows. We sequentially fix one edge and solve the MSTP such that the selected edge is in the solution. That is to say we solve the following m problems for each edge p:

$$\overline{d}_p=\min\sum_{q=1}^{m}\overline{c}_{pq}s_q \tag{19}$$

$$\text{subject to} \quad \mathbf{s}\in\mathcal{T} \tag{20}$$

$$s_p=1 \tag{21}$$

where $\mathbf{s}$ denotes solution vector consisting of s_q values.

Then for each edge we obtain the objective function values $\overline{d}_p$. These values, which are computed for each edge, can be interpreted as the cost of being in the solution. Then using the costs $\overline{d}_p$ we solve the following MSTP:

$$L(\alpha)=\min\sum_{p=1}^{m}(\overline{d}_p+\overline{b}_p)x_p \tag{22}$$

$$\text{subject to} \quad \mathbf{x}\in\mathcal{T} \tag{23}$$

The solution of the above problem yields a Gilmore and Lawler type lower bound to the QMSTP. The Lagrangian relaxation subproblem will be solved by inspection as proposed by Frieze and Yadegar [5] for the QAP case. Whenever $x_p^*=0$ let $y_{pq}^*=0$ for all $q=1,\ldots,m$. If $x_p^*=1$ then let $y_{pp}^*=1$ and let y_{pq}^* be the value of s_q in the solution to (19)–(21). Now we will prove the following result.

Proposition 2. (x^*,y^*) *satisfy* (3)–(7), (16).

Proof. First of all note that constraints (5)–(7) are satisfied by definition. Then it remains to show that (x^*,y^*) satisfy constraints (3), (4) and (16). In order to show that constraints (16) are satisfied, first consider the case $x_p=1$. Recall that y_{pq} are set according to the values of s_q variables in the solution of the problem given by (19)–(21). Hence there is at least one edge incident to vertex i as a result of the spanning tree, which contains edge (i,j), obtained by (19)–(21). Therefore, constraints (16) are satisfied for the case $x_p=1$. On the other hand, for the case $x_p=0$, y_{pq} are set to zero, then constraints (16) are satisfied.

Now it remains to show that both constraints (3) and (4) are satisfied. Whenever x_p equals to 1 in the solution of the problem defined by (22) and (23), then $s_p=1$ holds in the solution of the problem given by (19)–(21) and $(n-2)$ edges other than edge p, forming a spanning tree, are also selected. Then the setting of y_{pq} variables as defined above satisfy constraints (3) for $p=1,\ldots,m$, since there are $(n-1)$ edges in a spanning tree obtained in the solution of the problem defined by (19)–(21). On the other hand, when x_p equals to zero then all y_{pq} are set to zero and again constraints (3) are satisfied. Similarly, we can show that constraints (4) are satisfied. This completes the proof. □

Once the solution of the $L(\alpha)$ is obtained then a standard subgradient algorithm is employed to update the Lagrangian multipliers α in an iterative way.

4. A local search algorithm for the QMSTP

In this section we propose a neighborhood and a local search algorithm for the QMSTP. In the following subsection we will present the neighborhood definition and search scheme. Then, in the next subsection the local search algorithm will be discussed.

4.1. A neighborhood search scheme for the QMSTP

Before defining the neighborhood we will give the following definitions. A feasible solution (i.e. spanning tree) $\mathbf{t} \in \mathcal{T}$ is defined on a complete graph G. There are two sets of edges: *basic edges* and *nonbasic edges*. A *basic edge* is an edge in the current feasible solution $\mathbf{t} \in \mathcal{T}$. All other edges, which are not in the current tree, are defined as *nonbasic edges*.

The $N^k_{(t)}$ neighborhood is as follows. Randomly select k basic edges of the current spanning tree $\mathbf{t} \in \mathcal{T}$. Replace each basic edge $e_p \in \mathbf{t}$ with a nonbasic edge $e_q \notin \mathbf{t}$ such that a spanning tree is maintained at the end of this operation. The 1-edge neighborhood $N^1_{(t)}$ is illustrated with Fig. 1. In this example a five vertices graph is considered and edge (1,4) is randomly selected as the leaving edge. Then, edges (4,5), (2,4) and (3,4) are eligible to enter into the solution. The one which yields the best improvement will be selected in the new spanning tree. Here, edge (3,4) is inserted. The 2-edges neighborhood $N^2_{(t)}$ is illustrated with Fig. 2. Again a 5-vertex complete graph is considered and edges (1,4) and (1,3) are randomly selected as the edges to be replaced. Then edges (2,3),(2,4),(3,5),(3,4) and (4,5) are eligible to enter into the solution. Note that edge pairs (2,4)–(4,5) and (2,3)–(3,5) cannot be placed onto the solution since when selected they form a cycle rather than a spanning tree. In this example, edges (4,5) and (2,3) are inserted into the tree.

Clearly, the larger the number of removed edges from the current spanning tree, the wider the search space. In a local search algorithm it is possible to employ both $N^1_{(t)}$ and $N^2_{(t)}$ neighborhoods. Evidently, $N^2_{(t)}$ is more powerful than $N^1_{(t)}$, however, $N^2_{(t)}$ may require excessive CPU times to check all possible combinations. $N^1_{(t)}$ may be useful for the efficiency of the local search since it is faster than $N^2_{(t)}$. Hence, the choice of the neighborhood is a critical issue which affects both the efficiency and the accuracy of the search algorithm. Therefore, this critical decision must be carefully handled considering the size of the instances.

4.2. The local search algorithm

In this subsection we present a basic local search procedure which works quite efficiently for most of the QMSTP instances. Our local search algorithm was inspired on the tabu thresholding idea proposed by Glover [8]. The local search algorithm mainly consists of two phases. The first phase is a neighborhood search phase. The second phase consists of a random move from a local optimum. Before each random move, the best move is examined to see if it improves the best solution so far and if it does, the random move phase is terminated prematurely. The algorithm alternately runs the neighborhood search and the random move phases until a fixed iteration limit is reached.

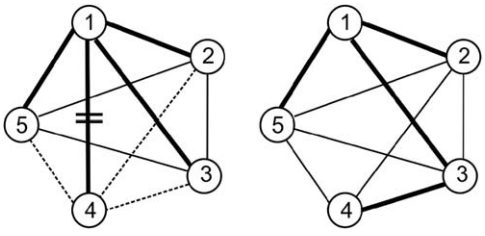

Fig. 1. $N^1_{(t)}$ neighborhood.

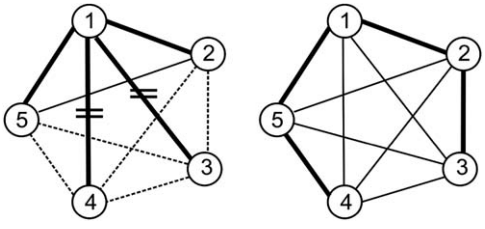

Fig. 2. $N^2_{(t)}$ neighborhood.

Algorithm 1. Random Local Search With Tabu Thresholding

begin
Input: l : degree of freedom,
$iter(l)$: iteration limit per degree of freedom
k : number of edges to be exchanged $N^k_{(t)}$
$count\_limit$: iteration limit of the local search
Output: The best solution found and its value
$\mathbf{t} \leftarrow$ A random spanning tree (a feasible solution);
Compute $f(\mathbf{t})$ (objective function value of $\mathbf{t}$)
$solution \leftarrow \mathbf{t}$; $value \leftarrow f(\mathbf{t})$; $count \leftarrow 1$;
repeat
while $\mathbf{t}$ is not a local minimum **do**
$f(\mathbf{t}_{pq}) \leftarrow \min\{f(\mathbf{t}_{rs}) : \mathbf{t}_{rs} \in N^k_{(t)}\}$;
$\mathbf{t} \leftarrow \mathbf{t}_{pq}$;
end
/* A local minimum reached.
update the best solution, if necessary */
if $value > f(\mathbf{t})$ **then** $solution \leftarrow \mathbf{t}$; $value \leftarrow f(\mathbf{t})$;
/* Random move starts */
$i \leftarrow 1$
repeat
$j \leftarrow$ A random integer in $[1, l]$;
$\mathbf{t}_{uv} \leftarrow j$th best element of $\{f(\mathbf{t}_{rs}) : \mathbf{t}_{rs} \in N^k_{(\mathbf{t})}\}$;
$\mathbf{t} \leftarrow \mathbf{t}_{uv}$; /* Random move */
$i \leftarrow i+1$;
$f(\mathbf{t}_{pq}) \leftarrow \min\{f(\mathbf{t}_{rs}) : \mathbf{t}_{rs} \in N^k_{(t)}\}$;
if $value > f(\mathbf{t}_{pq})$ **then**
$\mathbf{t} \leftarrow \mathbf{t}_{pq}$;
$solution \leftarrow \mathbf{t}_{pq}$;
$value \leftarrow f(\mathbf{t}_{pq})$;
$i \leftarrow iter(l)$;
/* Random move step terminated by
aspiration criterion */
end
until $i > iter(l)$;
$count \leftarrow count + 1$;
/* Random move phase is over. New local search
begins. */
until $count > count\_limit$;
Output solution value and STOP the algorithm
end

In the local search algorithm presented above, different neighborhood schemes can be employed. For the sake of efficiency we confined ourselves to employ only $N^1_{(t)}$ neighborhoods which seemed to yield good solutions in a reasonable CPU time. Recently, Kaveh and Punnen [10] have reported that the random local search algorithm with tabu thresholding performed well for the QAP. Hence, we can expect it to work well for the QMSTP.

The basic idea of the definition of $N^k_{(t)}$, which is well known and used by various researchers in developing local search algorithms for hard combinatorial optimization problems, is classified as k-exchange neighborhoods. $N^k_{(t)}$ is our adaptation of the k-exchange neighborhood for the QMSTP. Recently we are told that a similar neighborhood was used by Cordone and Passeri [4] to develop a Tabu search (TS) heuristic for the QMSTP. Our work is independent of theirs. While our experiments are based on Tabu thresholding,

Cordone and Passeri [4] consider TS. Obviously neither approach is uniformly superior to the other uniformly. Thus, as we initiated in this study, we restrict our experiments to Tabu thresholding only. For results on TS we refer to Cordone and Passeri [4].

5. Computational experiments

In this section, we report the computational experiments for both the lower bounding approaches and the local search algorithm. In the following subsection we present the generation of test instances. Then, we will discuss in detail the results of our computational results.

5.1. Generation of test instances

To the best of our knowledge, the only standard test instances for the QMSTP are due to Soak et al. [18,19]. However, their QMSTP test set consists of five instances. Hence, we have generated two groups of test instances. The first group consists of randomly generated test instances. The second group is obtained by transforming QAP test instances from the QAPLIB [2].

The first group consists of three sets of instances: Test Set I, Test Set II and Test Set III. Test Set I include symmetric test instances where the interaction costs a_{pq} are set equal to a_{qp} for $p,q=1,\ldots,m; p\neq q$. In this set, the interaction costs a_{pq} and edge costs b_p are chosen according to $U(1,20)$ and $U(1,100)$, respectively. Here $U(x,y)$ represents uniformly distributed random integers in the range $[x,y]$. For instances in Test Set II, a weight chosen according to $U(1,10)$ is associated with each vertex i for $i=1,\ldots,m$. The interaction costs a_{pq} are computed as a product of the weight values associated with edges e_p and e_q. For example, consider edges e_p and e_q connecting vertices (1,2) and (3,4), respectively. Then the value of a_{pq} is computed by multiplying the weights of vertices 1,2,3 and 4. Taking into consideration the scale of the interaction costs, b_p values are chosen according to $U(1,10\,000)$. To generate instances in Test Set III, first vertex coordinates are randomly chosen in the rectangle with coordinates (0,0), (0,100), (100,0) and (100,100). Then b_p values are computed as the distances between vertices connected by edge e_p. Last, the interaction costs a_{pq} are computed as the distance between the coordinates of the midpoints of edges e_p and e_q. For each test set, we have generated problems with size ranging from n=6 to 50. For each problem size of n=6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 20, 30 and 50 we have generated 10 randomly generated instances. Hence, each test set consists totally of $16\times 10=160$ instances. Therefore, the first group contains $160\times 3=480$ randomly generated test instances. All instances in the first group are complete graphs with n vertices and $m=n(n-1)/2$ edges.

The second group consists of transformed QAPLIB instances. Fig. 3 illustrates the transformation from a QAP instance in bipartite graph G_B into a QMSTP instance in complete graph G. In the left-hand side of Fig. 3 we have a bipartite graph with n pairs and in the right-hand side we have a complete graph with $2n$ vertices. As can be observed, with this transformation, we have added $\binom{n}{2}$ edges into the left-hand side and $\binom{n}{2}$ edges into the right-hand side of the bipartite graph. Let E_L and E_R denote the set of edges connecting vertices in the left-hand side and the right-hand side of the bipartite graph G_B, respectively. Let E_C denote the set of crossing edges in G_B.

Before giving in detail the interaction costs of the edges of the complete graph we would like to express the objective function of the QMSTP in the following explicit form:

$$\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\sum_{k=1}^{n}\sum_{\substack{l=1\\ l\neq k}}^{n} c_{ijkl}x_{ij}x_{kl}$$

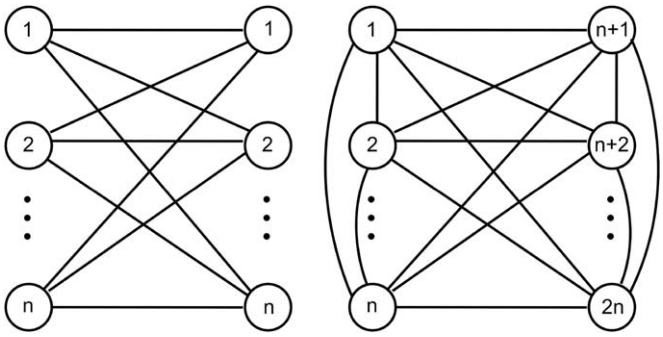

Fig. 3. Transformation from a QAP instance in bipartite graph into a QMSTP instance in complete graph.

where c_{ijkl} denotes the interaction cost when links from vertex i to vertex j and from vertex k to vertex l are established. Here the decision variable x_{ij} is equal to 1 if and only if vertex i is connected to vertex j. The interaction costs used in the proposed transformation process are set as follows.

First of all, the interaction costs of the edges connecting the left hand side vertices are penalized:

$$c_{ijkl}=\begin{cases}0 & \text{if } i=k \text{ and } j=l\\ M & \text{otherwise}\end{cases}$$
$$\text{for } i,j,k,l=1,\ldots,n,\ \ i<j,\ \ k<l \tag{24}$$

where M is a large number. The interaction costs between the edges E_L and E_C are also penalized:

$$c_{ijkl}=M \quad \text{for } i,j=1,\ldots,n,\ \ i<j,\ \ k=1,\ldots,n,\ \ l=n+1,\ldots,2n$$
$$c_{ijkl}=M \quad \text{for } i=1,\ldots,n,\ \ j=n+1,\ldots,2n,\ \ k,l=1,\ldots,n,\ \ k<l \tag{25}$$

Then, all interaction costs of the edges in E_L and E_R are penalized except the interaction costs of the edges $(i,i+1)$ for $i=n+1,\ldots,2n-1$ and the edges in E_L:

$$c_{ijkl}=M \quad \text{for } i,j=1,\ldots,n,\ \ k,l=n+1,\ldots,2n,\ \ k<l$$
$$c_{ijkl}=\begin{cases}0 & \text{if } i+1=j\\ M & \text{otherwise}\end{cases}$$
$$\text{for } i,j=n+1,\ldots,2n,\ \ i<j,\ \ k,l=1,\ldots,n,\ \ k<l \tag{26}$$

The interaction costs of all edges in E_C and the edges in E_R except $(k,k+1)$ for $k=n+1,\ldots,2n-1$ are penalized:

$$c_{ijkl}=\begin{cases}0 & \text{if } k+1=l\\ M & \text{otherwise}\end{cases}$$
$$\text{for } i=1,\ldots,n,\ \ j=n+1,\ldots,2n,\ \ k,l=n+1,\ldots,2n,\ \ k<l$$
$$c_{ijkl}=\begin{cases}0 & \text{if } i+1=j\\ M & \text{otherwise}\end{cases}$$
$$\text{for } i,j=n+1,\ldots,2n,\ \ i<j,\ \ k=1,\ldots,n,\ \ l=n+1,\ldots,2n \tag{27}$$

The interaction costs c_{ijkl} of the edges in E_C are set to $g_{ik}h_{jl}$ for $i\neq k$ and $j\neq l$ where g_{ik} and h_{jl} are the cost parameters of the QAP:

$$c_{ijkl}=\begin{cases}0 & \text{if } i=k,\ j=l\\ g_{ik}h_{j-n,l-n} & \text{if } i\neq k,\ j\neq l\\ M & \text{otherwise}\end{cases}$$
$$\text{for } i=1,\ldots,n,\ \ j=n+1,\ldots,2n,\ \ k=1,\ldots,n,\ \ l=n+1,\ldots,2n \tag{28}$$

Finally, the interaction costs in the right-hand side vertices of the bipartite graph are set:

$$c_{ijkl} = \begin{cases} 0 & \text{if } i=k, j=l \text{ or } k+1=l \\ M & \text{otherwise} \end{cases}$$
$$\text{for } i,j = n+1,\ldots,2n,\ \ i<j,\ \ k,l = n+1,\ldots,2n,\ \ k<l \qquad (29)$$

As can be observed, the optimum solution of the bipartite matching problem is untouched and this optimum solution is also optimum for the QMSTP on the transformed complete graph. Furthermore, note also that the edges $(i,i+1)$ for $i=n,\ldots,2n-1$ are necessarily in the optimum solution since their interaction costs with all other edges are set to 0.

Proposition 3. *The optimum value of the QMSTP instance obtained with this transformation is equal to the optimum value of the QAP instance, assuming that a solution of the corresponding QAP instance exists.*

Proof. First of all, note that the interaction costs of the edges connecting vertices $i=1,\ldots,n$ are set to M for all cases except their interaction costs with the edges $(i,i+1)$ for $i=n+1,\ldots,2n-1$. This setting, which is ensured by (24)–(26), forbids the inclusion of edges in E_L in the optimal solutions due to the relative value of M when compared with other interaction costs. The definition of costs stated in (27) enforce that edges $(i,i+1)$ for $i=n+1,\ldots,2n-1$ are necessarily in the optimal solution. By (29) the interaction costs of the edges $(i,i+1)$ for $i=n+1,\ldots,2n-1$ with other edges linking vertices $i=n+1,\ldots,2n$ are set to 0 and all other interaction costs of the edges connecting vertices $i=n+1,\ldots,2n$ are set to M. Finally, interaction costs between the crossing edges are set according to (28) which imply that the optimum solution of the QMSTP

Table 1
Computational experiments with relaxations of the AX2 and EFVI.

Number of vertices	AX2+(30)		EFVI+(30)		AX2+(33–39)		EFVI+(33–39)	
	LPR	CPU	LPR	CPU	LPR	CPU	LPR	CPU
6	178.20	0.0	217.58	0.1	179.10	0.01	217.61	0.01
7	192.08	0.0	252.56	0.2	197.35	0.01	252.64	0.05
8	231.52	0.0	325.67	0.2	232.65	0.02	325.67	0.21
9	267.87	0.0	371.13	0.2	269.13	0.03	371.13	0.36
10	280.84	0.02	407.95	0.5	281.48	0.04	407.95	0.98
11	286.78	0.02	447.01	0.9	286.90	0.06	447.01	2.58
12	291.91	0.02	479.93	2.3	293.16	0.11	479.93	6.24
13	318.94	0.06	523.36	6.7	320.03	0.22	523.36	10.45
14	337.77	0.06	588.97	9.9	339.00	0.33	588.97	23.09
15	318.72	0.08	588.73	22.3	319.89	0.59	588.73	48.72
16	337.59	0.13	638.13	39.9	337.77	0.74	638.13	61.38
17	350.91	0.22	687.91	63.1	352.10	1.42	687.91	130.88
18	374.66	0.31	739.16	111.9	375.47	1.71	739.16	246.59
20	373.77	0.63	766.04	352.7	374.09	6.61	794.49	477.28
Average	**295.82**	**0.1**	**502.44**	**43.6**	**297.01**	**0.85**	**504.48**	**72.06**
6	14269.62	0.0	15732.38	0.01	15025.86	0.0	15859.39	0.0
7	17961.12	0.0	19447.21	0.01	18781.71	0.01	19502.00	0.01
8	22367.49	0.0	26470.26	0.01	24487.18	0.02	26523.86	0.02
9	18802.15	0.01	22403.53	0.02	21241.87	0.03	22604.53	0.05
10	21884.20	0.01	25020.14	0.08	23926.08	0.08	25416.20	0.36
11	23928.39	0.02	28804.59	0.2	26915.27	0.12	28991.95	0.5
12	27756.08	0.02	32424.86	0.2	30251.24	0.17	32515.28	0.68
13	32337.28	0.04	39720.21	0.4	36066.86	0.24	39842.14	0.85
14	36658.45	0.05	44017.78	0.6	41014.33	0.38	44097.41	1.32
15	39974.06	0.08	50103.75	0.9	45986.50	0.59	50348.89	2.91
16	34249.49	0.09	41304.12	1.0	38437.75	1.14	41641.42	3.23
17	30342.95	0.11	41347.13	1.2	35916.24	2.48	41564.19	5.64
18	34380.47	0.16	45010.76	2.1	40321.67	3.4	45490.61	6.75
20	39358.42	0.48	55019.05	7.3	48590.76	5.8	55177.53	14.2
Average	**28162.16**	**0.1**	**34773.27**	**1.0**	**31925.95**	**1.03**	**34969.67**	**2.61**
6	346.53	0.0	512.68	0.01	355.59	0.0	516.49	0.01
7	442.10	0.01	734.54	0.02	456.25	0.01	737.23	0.03
8	529.08	0.02	936.33	0.05	546.54	0.02	942.54	0.2
9	770.09	0.02	1240.17	0.14	776.01	0.03	1240.33	0.4
10	716.83	0.02	1295.06	0.28	733.57	0.05	1296.16	1.4
11	744.24	0.02	1360.75	0.6	760.21	0.08	1369.34	1.9
12	888.76	0.03	1691.48	1.5	904.51	0.11	1692.01	4.6
13	1006.96	0.04	1930.95	2.3	1015.62	0.14	1934.91	9.9
14	1113.59	0.05	2225.79	5.3	1126.34	0.19	2232.50	16.3
15	1224.53	0.08	2264.80	15.5	1229.04	0.33	2267.37	48.2
16	1269.34	0.1	2555.67	33.8	1274.45	0.5	2563.74	81.1
17	1289.94	0.13	2914.60	75.7	1308.16	0.6	2921.00	160.7
18	1396.92	0.21	3015.51	126.2	1412.05	0.78	3022.95	205.7
20	1533.62	0.38	3516.39	183.7	1544.72	2.42	3518.10	279.16
Average	**948.04**	**0.1**	**1871.05**	**31.8**	**960.22**	**0.38**	**1875.33**	**57.83**

instance contains the edges which are in the optimum solution of the QAP instance plus the edges $(i, i+1)$ for $i = n+1, \ldots, 2n-1$. □

5.2. Computational results

We now present the details of our computational experiments. The algorithms are coded in C++ and tested on a Pentium IV PC with a 3 GHz CPU and 2 GB RAM. Linear programming (LP) and mixed integer linear programming (MILP) problems are solved by Cplex 11.0 with default options.

In Tables 1–3 we report computational experiments performed on randomly generated test sets. Table 1 includes the computational experiments with the relaxations of the AX2 and EFVI. Computational experiments with the lower bounding schemes and the local search algorithms are presented with Tables 2 and 3, respectively. Each of these tables is divided into three parts where we separately report results obtained with Test Set I, Test Set II and Test Set III. The last row of each part provides the average values of the corresponding columns. Each row of these tables gives the average values of the respective 10 randomly generated test instances. The first columns of these tables contain the number of vertices.

The first part of the computational results consists of the comparison of the AX2 and EFVI. To compare their strength we have solved their relaxations. It is possible to state that given two

Table 2
Computational experiments with lower bounding schemes.

Number of vertices	Gilmore-Lawler		Assad-Xu		Lagrangian Rel.		AX vs. GL	LR vs. GL
	LB	CPU(s)	LB	CPU(s)	LB	CPU(s)	% dev	% dev
6	147.5	0.0	230.0	0.0	258.4	0.11	55.9	75.2
7	167	0.0	273.9	0.0	312.9	0.17	64.0	87.4
8	209	0.0	361.2	0.0	409.8	0.37	72.8	96.1
9	232.2	0.0	408.6	0.0	505.6	0.79	76.0	117.7
10	240.3	0.0	463.5	0.0	547.9	1.27	92.9	128.0
11	250.4	0.0	505.8	0.0	613.2	1.95	102.0	144.9
12	256.1	0.0	557.7	0.0	652.1	2.62	117.8	154.6
13	282.4	0.0	608.5	0.0	713.0	3.88	115.5	152.5
14	300.0	0.0	686.2	0.0	812.1	5.90	128.7	170.7
15	285.1	0.0	692.3	0.0	860.1	7.34	142.8	201.7
16	299.2	0.0	757.4	0.0	937.1	14.2	153.1	213.2
17	324	0.0	817.4	0.0	1020.3	21.0	152.3	214.9
18	334.4	0.0	878.2	0.0	1085.5	24.0	162.6	224.6
20	339	0.0	960.8	0.1	1220.0	28.19	183.4	259.9
30	391.3	0.0	1399.7	0.2	1828.0	301.0	257.7	367.2
50	319.9	0.3	2015.3	1.6	2845.1	3311.7	530.0	789.4
Average	**273.6**	**0.0**	**726.0**	**0.1**	**913.8**	**232.8**	**150.5**	**212.4**
6	12335.7	0.0	16043.2	0	16290.4	0.1	30.1	32.1
7	15050.5	0.0	19425.9	0	19625.7	0.2	29.1	30.4
8	18735.4	0.0	26500.5	0	26739.4	0.4	41.4	42.7
9	17718.5	0.0	22593.8	0	22768.1	0.7	27.5	28.5
10	19438.4	0.0	25524.7	0	25750.5	1.2	31.3	32.5
11	21870.3	0.0	29015.1	0	29295.0	1.8	32.7	33.9
12	25074.5	0.0	32330	0	32607.2	2.6	28.9	30.0
13	29498.8	0.0	40054	0	40458.1	3.8	35.8	37.2
14	33743	0.0	43938.6	0	44310.1	5.2	30.2	31.3
15	38170.9	0.0	50433.9	0	50789.0	9.1	32.1	33.1
16	34746	0.0	41472	0	41907.2	12.1	19.4	20.6
17	34410	0.0	41383	0	41779.6	15.8	20.3	21.4
18	36838.7	0.0	45521.1	0.1	46372.3	18.4	23.6	25.9
20	44496.6	0.0	54813.5	0.1	55862.2	25.3	23.2	25.5
30	69127.4	0.0	78542.9	0.3	80177.6	178.5	13.6	16.0
50	109990	0.4	123562	2.26	126281.0	1983.6	12.3	14.8
Average	**35077.8**	**0.0**	**43197.1**	**0.2**	**43813.3**	**141.2**	**27.0**	**28.5**
6	471.8	0.0	525.0	0.0	541.1	0.1	11.3	14.7
7	675.8	0.0	746.4	0.0	782.7	0.3	10.4	15.8
8	887.0	0.0	968.8	0.0	1020.1	0.5	9.2	15.0
9	1168.1	0.0	1272.7	0.0	1345.7	0.8	9.0	15.2
10	1224.4	0.0	1335.9	0.0	1422.0	1.3	9.1	16.1
11	1324.6	0.0	1450.9	0.0	1535.6	1.8	9.5	15.9
12	1623.5	0.0	1760.8	0.0	1893.9	2.9	8.5	16.7
13	1861.1	0.0	2016.8	0.0	2169.2	3.7	8.4	16.6
14	2164.1	0.0	2335.9	0.0	2515.3	8.2	7.9	16.2
15	2237.9	0.0	2395.1	0.0	2578.2	7.3	7.0	15.2
16	2524.6	0.0	2726.0	0.0	2969.7	9.5	8.0	17.6
17	2837.4	0.0	3068.7	0.1	3372.1	12.6	8.2	18.8
18	2999.2	0.0	3241.9	0.2	3510.0	17.4	8.1	17.0
20	3471.3	0.0	3760.2	0.4	4329.8	22.6	8.3	24.7
30	6608.5	0.0	7107.7	1.2	8311.1	163.7	7.6	25.8
50	12806.8	0.4	13726.7	9.7	16127.0	1795.2	7.2	25.9
Average	**2805.4**	**0.0**	**3027.5**	**0.7**	**3401.5**	**128.0**	**8.6**	**18.0**

Table 3
Computational experiments with the local search algorithm.

Number of vertices	Assad-Xu		Assad-Xu		LS	CPU(s)	Aver	Max	Min	Gap	No. of opt. solutions	
	H1	CPU(s)	H2	CPU(s)			Imp	Imp	Imp		AX	LS
6	282.6	0.0	264.1	0.0	258.4	0.2	2.2	11.7	0.00	0.0	6	10
7	354.7	0.0	337.2	0.0	326.8	0.4	3.1	11.4	0.00	4.4	3	10
8	507.4	0.0	461.2	0.0	438.5	1.2	4.9	9.4	0.00	7.0	2	10
9	645.4	0.0	559	0.0	534.9	2.1	4.3	10.5	0.00	6.2	2	10
10	784.9	0.0	679.9	0.0	650.1	3.3	4.4	8.3	0.00	18.0	1	10
11	976.9	0.0	817.9	0.0	785.9	4.7	3.9	10.8	0.00	28.4	*	*
12	1107.9	0.0	947	0.0	915.1	7.2	3.4	11.4	0.00	40.8	*	*
13	1365	0.0	1127	0.0	1053.33	12.2	6.5	9.0	0.00	50.2	*	*
14	1585.6	0.0	1305.8	0.0	1235.17	16.6	5.4	8.7	0.00	54.3	*	*
15	1781	0.0	1467.8	0.0	1358.75	29.5	7.4	10.7	0.20	61.8	*	*
16	2073.5	0.0	1752.7	0.0	1594.94	60.2	9.0	10.6	1.70	74.1	*	*
17	2296.7	0.0	1923.7	0.0	1801.78	74.3	6.3	9.8	1.61	79.1	*	*
18	2640.5	0.0	2230.9	0.0	2036.68	90.4	8.7	10.2	1.75	92.4	*	*
20	3313.8	0.0	2728	0.0	2525.01	121.4	7.4	10.1	1.40	111.2	*	*
30	7937	0.0	6396.5	0.1	5929.36	511.1	7.3	9.3	1.80	230.3	*	*
50	22792.4	0.0	18513.6	1.2	18154.5	2615.2	1.9	3.1	1.10	815.3	*	*
Average	**3152.8**	**0.0**	**2594.5**	**0.1**	**2475.0**	**231.7**	**5.4**	**9.7**	**0.8**	**104.6**	**2.8**	**10**
6	18187.4	0.0	16449.3	0.0	16290.4	0.3	1.0	4.8	0.0	0.0	8	10
7	23040.3	0.0	19700.9	0.0	19625.7	0.4	0.4	3.5	0.0	0.0	7	10
8	32952.3	0.0	27371.9	0.0	27039.4	1.4	1.2	6.0	0.0	0.0	5	10
9	26163.6	0.0	22823.1	0.0	22769.9	2.6	0.2	2.1	0.0	0.0	6	10
10	28256.3	0.0	25931.7	0.0	25750.5	3.9	0.7	2.6	0.0	0.0	5	10
11	33513.6	0.0	29615.1	0.0	29307.6	4.8	1.0	3.3	0.0	0.0	*	*
12	40345.9	0.0	32728.9	0.0	32615.8	8.4	0.3	1.5	0.0	0.0	*	*
13	51446.5	0.0	41288.1	0.0	40488.5	13.5	1.9	5.0	0.0	0.1	*	*
14	55575.1	0.0	44814.9	0.0	44338.2	26.3	1.1	3.6	0.0	0.1	*	*
15	67349.4	0.0	52348.3	0.0	50821.6	32.5	2.9	4.8	0.0	0.1	*	*
16	49602	0.0	42397.2	0.0	41940.2	44.1	1.1	3.4	0.0	0.1	*	*
17	49209.5	0.0	42647.6	0.0	41819.6	57.3	1.9	6.0	0.2	0.1	*	*
18	54614.8	0.0	46939.9	0.0	45145.2	70.4	3.8	5.6	2.1	0.9	*	*
20	67991.2	0.0	56990.9	0.0	54610.3	124.1	4.2	5.4	2.3	1.6	*	*
30	88605.1	0.0	81676.2	0.1	78999.9	523.3	3.3	7.1	1.4	2.5	*	*
50	180358	0.0	172391	1.2	164040	2380.4	4.8	5.5	3.1	3.7	*	*
Average	**54200.7**	**0.0**	**47257.2**	**0.1**	**45975.1**	**205.9**	**1.9**	**4.4**	**0.6**	**0.6**	**6.2**	**10.0**
6	587.7	0.0	551.8	0.0	541.2	0.2	1.9	11.8	0.0	0.0	7	10
7	833.9	0.0	803.4	0.0	783.7	0.6	2.5	11.9	0.0	0.1	6	10
8	1051.2	0.0	1030.2	0.0	1020.1	1.6	1.0	3.6	0.0	0.0	6	10
9	1436	0.0	1397.8	0.0	1356	2.1	3.0	8.5	0.0	0.8	2	10
10	1536.6	0.0	1500.8	0.0	1427.1	4.8	4.9	15.6	0.0	0.4	1	10
11	1643.5	0.0	1589.7	0.0	1545.1	7.9	2.8	14.5	0.0	0.5	*	*
12	1970.5	0.0	1960	0.0	1901.6	9.6	3.0	9.5	0.0	0.4	*	*
13	2250.7	0.0	2206.2	0.0	2175.3	14.1	1.4	6.4	0.0	0.3	*	*
14	2707.2	0.0	2622.9	0.0	2527.9	21.2	3.6	14.8	0.0	0.5	*	*
15	2709.6	0.0	2651.8	0.0	2588.8	25.7	2.4	8.3	0.0	0.4	*	*
16	3127.6	0.0	3010	0.0	2980.1	33.9	1.0	4.0	0.0	0.3	*	*
17	3688.1	0.0	3554.9	0.0	3372.2	47.2	5.1	15.2	0.0	0.0	*	*
18	3844.1	0.0	3797.5	0.0	3646.9	69.3	4.0	12.5	0.5	1.9	*	*
20	4494.4	0.0	4427.3	0.0	4193.8	102.3	5.3	9.7	0.6	1.7	*	*
30	8492.6	0.0	8351.5	0.1	7992.7	515.1	4.3	9.2	0.7	2.7	*	*
50	16885.9	0.0	16328.8	1.5	15654.5	2424.1	4.1	7.3	2.3	9.0	*	*
Average	**3578.7**	**0.0**	**3486.5**	**0.1**	**3356.7**	**205.0**	**3.1**	**10.2**	**0.3**	**1.2**	**4.4**	**10.0**

formulations of a minimization problem, the one yielding the larger relaxation value is better. Different formulations of a given problem can be frequently stated in terms of different sets of variable as in the case with the MSTP. In their early work, Magnanti and Wolsey [13] have discussed several alternative MSTP formulations. Among them the ones based on *packing* constraints and multicommodity flow (MCF) constraints are two well known formulations. Therefore, one way to replace constraints $\mathbf{x} \in \mathcal{T}$ in the AX2 and EFVI is to employ the packing constraint set:

$$\sum_{p=1}^{m} x_p = n-1 \tag{30}$$

$$\sum_{e_p \in S} x_p \le |S|-1, \quad S \subset V, \ 2 \le |S| \le n-1 \tag{31}$$

$$x_p \in \{0,1\}, \quad p = 1,\ldots,m \tag{32}$$

Constraints (30) ensure that the tree exactly consists of $n-1$ edges. Inequalities (31) are known as the packing constraints and state that the number of edges which can be packed in the set of vertices S cannot exceed $|S|-1$. Unfortunately, there are $\mathcal{O}(2^n)$ packing inequalities which can be gradually added to the formulation whenever they are violated.

Another way to replace constraints $\mathbf{x} \in \mathcal{T}$ in the AX2 and EFVI is to employ the MCF constraints [13]. The MCF constraints are

defined on a directed graph $G'=(V,A)$ obtained from $G=(V,E)$ where $A=\{(i,j): i,j\in V\}$ is the arc set. In $G'=(V,A)$, each undirected edge $e_p\in E$ is replaced by two directed arcs (i,j) and (j,i) where $i\neq j$ with symmetric costs. In the MCF model, n commodities are sent from a given root vertex r, which serves as a source, to other vertices k with unit demands. The flow variable w_{ij}^k is equal to one if and only if the commodity going from the root vertex r to vertex k flows on arc (i,j). The u_{ij} variable can be interpreted as capacity bound on arc (i,j). The MCF constraints are defined as follows

$$\sum_{\substack{i=1\\ i\neq j}}^{n} w_{ij}^k - \sum_{\substack{i=1\\ i\neq j}}^{n} w_{ji}^k = 0 \quad \text{for } j,k=1,\ldots,n,\ j\neq k,\ j\neq r \tag{33}$$

$$\sum_{\substack{j=1\\ j\neq r}}^{n} w_{jr}^k - \sum_{\substack{j=1\\ j\neq r}}^{n} w_{rj}^k = -1 \quad \text{for } k=1,\ldots,n,\ k\neq r \tag{34}$$

$$\sum_{\substack{j=1\\ j\neq k}}^{n} w_{jk}^k - \sum_{\substack{j=1\\ j\neq k}}^{n} w_{kj}^k = 1 \quad \text{for } k=1,\ldots,n,\ k\neq r \tag{35}$$

$$w_{ij}^k \le u_{ij} \quad \text{for every arc } (i,j) \text{ and } k=1,\ldots,n,\ k\neq r \tag{36}$$

$$u_{ij}+u_{ji}=x_p \quad \text{for } p=1,\ldots,m \tag{37}$$

$$\sum_{p=1}^{m} x_p = n-1 \tag{38}$$

$$w_{ij}^k \ge 0 \quad \text{for all arcs } (i,j)\in A \text{ and } k=1,\ldots,n \tag{39}$$

$$x_p\in\{0,1\} \quad \text{for } p=1,\ldots,m \tag{40}$$

Constraints (33)–(35) ensure the commodity flow from the root vertex r to other vertices k. Constraints (36) state that the commodity flow takes place for only the arcs (i,j) which are part of the directed tree from the root vertex r. The constraints (37) indicate that when a commodity flows from i to j or from j to i then the edge between i and j, corresponding to edge p, is included by the spanning tree.

In order to compare the strength of AX2 and EFVI, we consider two versions of relaxations. In the first relaxation version, we replace constraints $\mathbf{x}\in\mathcal{T}$ of the AX2 and EFVI with constraints (30). In the second relaxation version, we replace constraints $\mathbf{x}\in\mathcal{T}$ of the AX2 and EFVI with constraints (33)–(39). Computational experiments with the relaxations of the AX2 and EFVI on Test Set I, Test Set II and Test Set III are presented in Table 1. We have only reported results for instances with size up to 20 vertices because of excessive CPU time requirements in solving large sized LP problems. From the second to the fifth columns we report the lower bounds and CPU times in seconds obtained with the first relaxation versions of the AX2 and EFVI. From the sixth to the ninth columns are for the lower bounds obtained with the second relaxation versions of the AX2 and EFVI. Considering all test instances in Test Set I, Test Set II, and Test Set III, the overall average relative percent improvements obtained with the first (second) relaxation version of the EFVI over the first (second) relaxation version of the AX2 is 58.74 (53.63). Relative percent improvement of a lower bound LB_1 over another lower bound LB_2 is computed as follows $100\times(z_{LB_1}-z_{LB_2})/z_{LB_2}$. Hence, we can say that in average the relaxation of the EFVI yields tighter lower bounds than the relaxation of the AX2 does. These results show the strength of valid inequalities (15) and (16) since the EFVI is obtained by their addition to the AX2 formulation. In Appendix, we present an illustrative example which shows the relationship between the first relaxation versions of the AX2 and EFVI.

Furthermore, we have also solved the first and the second relaxation versions of the AX2 and EFVI by adding the integrality constraints (32). To obtain MILP bounds with the first relaxation versions of the AX2 and EFVI, we replace constraints $\mathbf{x}\in\mathcal{T}$ with constraints (30) and (32). To obtain MILP bounds with the second relaxation versions of the AX2 and EFVI, we replace constraints $\mathbf{x}\in\mathcal{T}$ with constraints (33)–(40). Unfortunately, we could only performed experiments for instances with sizes only up to $n=9$ vertices because of excessive CPU time requirements. The average percent deviation of LP bounds over MILP bounds for the first relaxation versions of AX2 (EFVI) are 43.16% (25.53%), 4.13% (0.55%) and 25.8% (4.18%) on Test Set I, Test Set II and Test Set III, respectively. The average percent deviation of LP bounds over MILP bounds for the second relaxation versions of the AX2 (EFVI) are 44.78% (26.49%), 7.06% (1.44%) and 42.74% (7.10%) on Test Set I, Test Set II and Test Set III, respectively. Considering the overall average of these percent deviations, which are 34.99%, 3.29% and 19.95% on Test Set I, Test Set II and Test Set III, respectively, we can say that Test Set I is the hardest test set and Test Set II arises to be the easiest one. The formulae used to compute these gaps is $100\times(z_{MILP}-z_{LP})/z_{MILP}$, where z_{MILP} and z_{LP} stand for the MILP and LP bounds, respectively.

Table 2 includes computational results on Test Set I, Test Set II and Test Set III, obtained with the lower bounding approaches. From the second to the seventh columns we present the average lower bound values and average CPU times obtained with the Gilmore and Lawler type lower bounding approach, Assad and Xu's leveling procedure and the Lagrangian relaxation approach. The last two columns give the percent improvements of the Assad and Xu's leveling procedure and the Lagrangian relaxation approach over the Gilmore and Lawler type lower bounding scheme, respectively. Considering all instances in Test Set I, Test Set II and Test Set III, the overall average percent improvement of the Assad and Xu's leveling procedure over the Gilmore and Lawler type lower bounding approach is 62% while the overall average percent improvement of the Lagrangian relaxation approach over the Gilmore and Lawler type lower bounding approach is 86.3%. From these results we can conclude that at the expense of the increase in CPU time requirement, the Lagrangian relaxation approach yields tighter lower bound values than the ones obtained by the Assad and Xu's leveling procedure.

Table 3 presents computational experiments with the local search algorithm on Test Set I, Test Set II and Test Set III. We have compared the proposed local search algorithms with our implementation of the heuristic algorithms H1 and H2 proposed by Assad and Xu [1]. The heuristic algorithm H1 first estimates the average contribution of each edge to the objective function. Then it solves the MSTP by using these estimate edge costs. The heuristic algorithm H2 selects the minimum weight edges in a greedy way and sequentially fixes them into the solution. The edge weights are updated after each selection. As pointed out by Assad and Xu [1], H2 is more complicated than H1. In Table 3 from the second to the seventh columns, we report the average upper bound values and average CPU times obtained with H1, H2 and the local search heuristic. Note that the solution values reported in the sixth column are obtained with single run of the local search algorithm. The eighth column denotes the average improvement obtained with the local search algorithm over the best solution obtained with H1 and H2. The ninth and tenth columns, respectively, stand for the maximum improvements and minimum improvements obtained with the local search algorithm over the best solution obtained with H1 and H2. To compute the improvement obtained with the local search algorithm over the best solution obtained with H1 and H2 we have employed the following formulae

$(\min\{z_{H_1}, z_{H_2}\} - z_{LS})/\min\{z_{H_1}, z_{H_2}\}$ where z_{H_1}, z_{H_2},and z_{LS} stand for the upper bound values obtained with H1, H2 and the local search heuristic, respectively. In the 11th column we give the relative gap between the local search algorithm and the Lagrangian relaxation scheme. The local search algorithm gives the lowest upper bound values and the Lagrangian relaxation scheme outputs the largest lower bound values, in all instances. The formulae used in the 11th column is $(z_{LS} - z_{LR})/z_{LR}$. The average gaps give a clue about the difficulty of our test sets. Since the average gap obtained with Test Set I is the highest (i.e., 104.6%) we can say that it is the most difficult test set. Notice that this column indicates the difficulty of the Test Sets. In the right-most two columns of Table 3 we report the number of times the optimum solution is reached with H1, H2 and the local search algorithm, respectively. To compute the optimum solutions of the randomtest instances we have solved a MILP formulation of the QMSTP which consists of (2)–(6) and the MCF constraints (33)–(40). These two columns help us to explain why the minimum improvements obtained with the local search algorithm over the best solution obtained with H1 and H2 are 0 for instances with sizes from $n=6$ to 10. Unfortunately, we were only able to optimally solve QMSTP instances with sizes up to $n=10$. An asterisk symbol ('*') is used for the unavailable results.

Experiments with the local search algorithm on the transformed instances from the QAPLIB with known optimal solution and on the instances proposed by Soak et al. [18,19] are presented with Tables 4 and 5, respectively. We report in Table 4 computational experiments with the local search algorithm, on some instances transformed from the QAPLIB with known optimal solution. Although we were not able to solve all QAPLIB instances due to the implications of their dimensions on the complexity of the solving procedure we believe that the results that we report will be sufficient to give an idea on the power of our local search algorithm. The first columns of these tables indicate the instances. We have used the original name of the QAP instance and added `Qmstp` at the end. The second column stands for the optimum value. The third and fourth columns indicate the upper bound values and CPU times obtained with H2, respectively. The fifth and sixth columns denote the upper bound values and CPU times obtained with our local search algorithm, respectively. The upper bound values reported in the fifth column are obtained with single run of the local search algorithm. The seventh columns indicate the relative deviation of the upper bounds obtained with H2 from the optimum solution values. The last two columns stand for the relative percent deviation of the upper bound values obtained with the local search from the optimum solution values. The formula used for the last two columns are $(z_{H_2} - z_{OPT})/z_{OPT}$ and

Table 5
Computational experiments with the local search algorithm on the instances provided by Soak et al. [18,19].

Size	Best	Min	Avg	Max	CPU(s)
50	25226	**0.00**	1.01	2.84	3242.1
60	35754	**0.00**	0.93	2.12	4321.4
70	48396	0.29	1.26	2.06	5738.5
80	63546	**0.00**	1.48	2.75	7026.3
90	79627	0.37	1.67	3.01	8623.6
100	98271	0.55	2.33	4.72	10431.3
Average		**0.20**	**1.45**	**2.92**	**6563.87**

Table 4
Computational experiments with the local search algorithm on the transformed instances from the QAPLIB with known optimal solution.

Instance	OPT.	H2		LS		H2 vs. OPT	LS vs. OPT	LS vs. H2
		UB	CPU(s)	UB	CPU(s)	% dev	%dev	%dev
nug12Qmstp	578	738	0.01	605	639	27.68	4.67	18.02
nug14Qmstp	1014	1366	0.1	1084	724	34.71	6.90	20.64
nug15Qmstp	1150	1486	0.1	1265	1348	29.22	10.00	14.87
nug16aQmstp	1610	2080	0.1	1742	2311	29.19	8.20	16.25
nug16bQmstp	1240	1580	0.1	1350	2936	27.42	8.87	14.56
nug17Qmstp	1732	2210	0.2	1874	3422	27.60	8.20	15.20
nug18Qmstp	1930	2518	0.2	2056	3482	30.47	6.53	18.35
nug20Qmstp	2570	3310	0.3	2860	5151	28.79	11.28	13.60
nug21Qmstp	2438	3380	0.4	2698	5184	38.64	10.66	20.18
nug22Qmstp	3596	5058	0.5	3868	5482	40.66	7.56	23.53
nug24Qmstp	3488	4548	0.9	3874	5914	30.39	11.07	14.82
nug25Qmstp	3744	4982	1.4	4083	5983	33.07	9.05	18.04
nug27Qmstp	5234	7022	1.8	5966	6025	34.16	13.99	15.04
nug28Qmstp	5166	6860	3.2	5819	6087	32.79	12.64	15.17
nug30Qmstp	6124	7824	3.8	6923	6227	27.76	13.05	11.52
Average	**2774.3**	**3664.1**	**0.9**	**3071.1**	**4061.0**	**31.5**	**9.5**	**16.7**
chr12aQmstp	9552	37370	0.01	11170	783	291.23	16.94	70.11
chr12bQmstp	9742	54502	0.01	10753	790	459.45	10.38	80.27
chr12cQmstp	11156	36960	0.01	12712	783	231.30	13.95	65.61
chr15aQmstp	9896	42726	0.1	11638	1239	331.75	17.60	72.76
chr15bQmstp	7990	53502	0.1	10145	1136	569.61	26.97	81.04
chr15cQmstp	9504	30662	0.1	12769	1254	222.62	34.35	58.36
chr18aQmstp	11098	48450	0.2	12757	3325	336.57	14.95	73.67
chr18bQmstp	1534	3626	0.2	1676	3354	136.38	9.26	53.78
chr20aQmstp	2192	8128	0.3	2445	4968	270.80	11.54	69.92
chr20bQmstp	2298	6360	0.3	2730	4652	176.76	18.80	57.08
chr20cQmstp	14142	114480	0.3	30124	4763	709.50	113.01	73.69
chr22aQmstp	6156	12798	0.5	8760	5089	107.89	42.30	31.55
chr22bQmstp	6194	13908	0.5	8402	4741	124.54	35.65	39.59
chr25aQmstp	3796	17448	1.2	9658	5223	359.64	154.43	44.65
Average	**7517.9**	**34351.4**	**0.3**	**10409.9**	**3007.1**	**309.1**	**37.2**	**62.3**

$(z_{LS}-z_{OPT})/z_{OPT}$, respectively. z_{OPT} denotes the optimal solution value. Note that, in Table 4 we have not reported the solution values obtained with H1 since in all those instances H1 yielded worse solution values than *M*. The last rows of Table 4 include the average values of the corresponding columns. As can be observed the average relative deviation of the upper bound obtained with H2 from the optimum solution value is 31.5% for the instances by Nugent et al. [15] and 309.1% for the instances by Christofides and Benavent [3]. However, the average relative deviation of the upper bounds obtained with the local search algorithm from the optimum solution value is 9.5% for the instances by Nugent et al. [15] and 37.2% for the instances by Christofides and Benavent [3]. In the first test set the local search algorithm yields three times more accurate upper bound values than H2. In the second test set the local search algorithm outputsalmost eight times more accurate upper bound values than H2, at the expense of increasing CPU time requirement.

Computational experiments with the local search algorithm on the instances provided by Soak et al. [18,19] are reported in Table 5. The first column stands for the size of the instances. The second column is for the best solution values found by Soak et al. [18,19]. The third, fourth and fifth columns include minimum, average and maximum values obtained with 20 independent runs of the local search algorithm, respectively. The last column includes the average CPU time of 20 runs. The last row gives the average values of the corresponding columns. As can be observed the local search algorithm reaches the best known solution values in three out of five instances. Hence we can say that the local search algorithm yields comparable performance to the EA proposed by Soak et al. [18,19].

Finally, note that the choice of the parameters is a critical step for the performance of the local search algorithm. After careful experimentation and fine-tuning, we can empirically say that setting $count\_limit := 1000$, $k := m$ and $iter(k) := 20 \times n$ seems to be a suitable choice for most of the instances.

6. Conclusion

We have developed a Lagrangian relaxation procedure and an efficient local search algorithm. We have also devised a new neighborhood scheme for the QMSTP and employed it in the local search algorithm. Furthermore, we have proposed a transformation scheme in order to test the performance of our local search algorithm on the QAPLIB instances with known optimum solution values.

Experimental results are reported on standard test instances, randomly generated test instances and QAP instances from the QAPLIB by using a transformation scheme. The proposed Lagrangian relaxation scheme yields the tightest lower bounds for all instances when compared to previous bounding approaches. The performance of the local search algorithm is very encouraging. We believe that combining the local search algorithm with different types of neighborhoods within a variable neighborhood search (Hansen and Mladenović, [9]) framework could provide improved solutions.

Acknowledgments

The first author acknowledges the support by TÜBİTAK Research Grant 107M462 and Galatasaray University Research Foundation Grant 09.402.005. The work of the second author was supported by an NSERC discovery grant. Thanks are due to two referees for their valuable comments which have helped improve the paper.

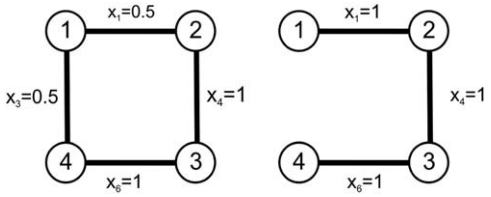

Fig. 4. Solutions obtained with the AX2 and EFVI on a test instance.

Appendix

We present a four vertex symmetric QMSTP test instance to show the relationship between the first relaxation version of the AX2 and EFVI. The cost matrix is as follows:

32	92	17	5	0	2
92	61	2	4	12	12
17	2	12	12	15	6
5	4	12	49	18	5
0	12	15	18	52	11
2	12	6	5	11	22

The numbers in the diagonal of the cost matrix correspond to edge costs and off-diagonal numbers denote the interaction costs. The first relaxation version of the AX2 yields the solution $y_{14}=y_{41}=y_{36}=y_{63}=y_{46}=y_{64}=1$, $x_4=x_6=1$, $x_1=x_3=0.5$, $y_{11}=y_{33}=0.5$ and $y_{44}=y_{66}=1$ with the objective function value 77. The solution is illustrated in the left-hand side of Fig. 4. However, when we add either constraints (15) or constraints (16) to the first relaxation version of the AX2 we obtain the solution $y_{14}=y_{16}=y_{34}=y_{36}=0.5$, $y_{41}=y_{46}=y_{63}=y_{64}=1$, $y_{11}=y_{33}=0.5$, $y_{44}=y_{66}=1$, $x_4=x_6=1$ and $x_1=x_3=0.5$ with the objective function value 78.5. Note that this solution is the same as the one we have obtained with the first relaxation version of the AX2 but with a higher objective function value. On the other hand, when we solve the first relaxation version of the EFVI, namely the first relaxation version of AX2 with both constraints (15) and (16), we obtain the following solution $y_{14}=y_{16}=y_{41}=y_{46}=y_{61}=y_{64}=1$, $x_1=x_4=x_6=1$ and $y_{11}=y_{44}=y_{66}=1$ with the objective function value 79. This solution, which is a spanning tree, is illustrated in the right hand side of Fig. 4.

References

[1] Assad A, Xu W. The quadratic minimum spanning tree problem. Naval Research Logistics 1991;39:399–417.
[2] Burkard RE, Karisch SE, Rendl F. QAPLIB—a quadratic assignment problem library. Journal of Global Optimization 1997;10:391–403.
[3] Christofides N, Benavent E. An exact algorithm for the quadratic assignment problem. Operations Research 1989;37(5):760–8.
[4] Cordone R, Passeri G. Heuristic and exact approaches for the quadratic minimum spanning tree problem. In: CTW 2008 seventh cologne twente workshop on graphs and combinatorial optimization. Italy: Gargano, University of Milan; 2008. p. 52–5.
[5] Frieze AM, Yadegar J. On the quadratic assignment problem. Discrete Applied Mathematics 1982;5:89–98.
[6] Gao J, Lu M. Fuzzy quadratic minimum spanning tree problem. Applied Mathematics and Computation 2005;164:773–88.
[7] Gilmore PC. Optimal and suboptimal algorithms for the quadratic assignment problem. SIAM Journal of Applied Mathematics 1962;10:305–13.
[8] Glover F. Tabu thresholding: improved search by nonmonotonic trajectories. ORSA Journal on Computing 1995;7:426–42.
[9] Hansen P, Mladenović N. Variable neighborhood search: principles and applications. European Journal of Operational Research 2001;130:449–67.
[10] Kaveh A, Punnen AP. Randomized local search and improved solutions for microarray layout QAP. Research Report, SFU Surrey, Department of Mathematics; 2008.
[11] Konno H, Kuno T. Linear multiplicative programming. Mathematical Programming 1992;56:51–64.

[12] Kruskal JB. On the shortest spanning subtree of a graph and the traveling salesman problem. Proceedings of the American Mathematical Society 1956;7(1):48–50.
[13] Magnanti TL, Wolsey LA. Optimal trees. In: Ball MO, Magnanti TL, Monma CL, Nemhauser GL, editors. Network Models, Handbook in Operations Research and Management Science, vol. 7. North-Holland: Amsterdam; 1995. p. 503–615.
[14] Lawler EL. The quadratic assignment problem. Management Science 1963;19:586–90.
[15] Nugent CE, Vollman TE, Ruml J. An experimental comparison of techniques for the assignment of facilities to locations. Operations Research 1968;16:150–73.
[16] Punnen AP. Combinatorial optimization with multiplicative objective function. International Journal of Operations and Quantitative Management 2001;3:222–7.
[17] Punnen AP. Nair KPK. On linear multiplicative programming. Opsearch 1997;34:139–54.
[18] Soak S-M, Corne DW, Ahn B-H. A new evolutionary algorithm for spanning-tree based communication network design. IEICE Transactions on Communication 2005;E88-B(10):4090–3.
[19] Soak S-M, Corne DW, Ahn B-H. The edge-window-decoder representation for tree-based problems. IEEE Transactions on Evolutionary Computation 2006;10:124–44.
[20] Xu W. Quadratic minimum spanning tree problems and related topics. PhD dissertation, University of Maryland; 1984.
[21] Zhou G, Gen M. An effective genetic algorithm approach to the quadratic minimum spanning tree problem. Computers and Operations Research 1998;25:229–37.