Discrete Optimization 8 (2011) 191–205

Contents lists available at ScienceDirect

Discrete Optimization

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

The minimum spanning tree problem with conflict constraints and its variations☆

Ruonan Zhang [a], Santosh N. Kabadi [b], Abraham P. Punnen [a,*]

[a] Department of Mathematics, Simon Fraser University Surrey, Central City, 250-13450 102nd AV, Surrey, British Columbia, V3T 0A3, Canada

[b] Faculty of Business Administration, University of New Brunswick, Fredericton, New Brunswick, Canada

ARTICLE INFO

Article history:
Received 12 August 2009
Received in revised form 6 May 2010
Accepted 12 August 2010
Available online 3 September 2010

Keywords:
Minimum spanning tree
Matroid intersection
Conflict graphs
Heuristics
Combinatorial optimization

ABSTRACT

We consider the minimum spanning tree problem with conflict constraints (MSTC). The problem is known to be strongly NP-hard and computing even a feasible solution is NP-hard. When the underlying graph is a cactus, we show that the feasibility problem is polynomially bounded whereas the optimization version is still NP-hard. When the conflict graph is a collection of disjoint cliques, (equivalently, when the conflict relation is transitive) we observe that MSTC can be solved in polynomial time. We also identify other special cases of MSTC that can be solved in polynomial time. Exploiting these polynomially solvable special cases we derive strong lower bounds. Also, various heuristic algorithms and feasibility tests are discussed along with preliminary experimental results. As a byproduct of this investigation, we show that if an ϵ-optimal solution to the maximum clique problem can be obtained in polynomial time, then a $(3\epsilon - 1)$-optimal solution to the maximum edge clique partitioning (Max-ECP) problem can be obtained in polynomial time. As a consequence, we have a polynomial time approximation algorithm for the Max-ECP with performance ratio $O\left(\frac{n(\log\log n)^2}{\log^3 n}\right)$, improving the best previously known bound of $O(n)$.

© 2010 Elsevier B.V. All rights reserved.

1. Introduction

The minimum spanning tree problem (MST) is perhaps the most well-solved combinatorial optimization problem. While MST can be solved in polynomial time by a greedy algorithm, many of its variations such as the Steiner tree problem [1], degree constrained minimum spanning tree problem [2], capacitated minimum spanning tree problem [3] etc. are NP-hard. Recently, Darmann et al. [4] introduced yet another variation of MST called the *minimum spanning tree problem with conflict pairs* (MSTC) which is the primary topic of discussion in this paper. See also [5] for additional results on MSTC and related problems. The feasibility version of MSTC was also introduced independently in [6,7] in the context of quadratic bottleneck spanning tree problem (QBSTP).

Let $G = (V, E)$ be an undirected graph with $|V| = n$ and $|E| = m$. For each edge $e \in E$ a cost c_e is prescribed. We are also given a set S of some two-element subsets of E. This set S is called the *conflict set* (conflict relation) and each $\{e, f\} \in S$ is called a *conflict pair*. A spanning tree T is called *conflict free* if T contains at most one edge from each conflict pair in S. Then, the problem MSTC is to find a least cost conflict free spanning tree of G.

Most of the minimum spanning tree applications have a natural interpretation in the presence of conflict pairs. The MSTC also includes as a special case the well-studied Hamiltonian path problem on a directed graph [8]. To see this,

☆ This work was supported by NSERC discovery grants awarded to Santosh N. Kabadi and Abraham P. Punnen.

* Corresponding author.
E-mail addresses: rza1@sfu.ca (R. Zhang), kabadi@unb.ca (S.N. Kabadi), apunnen@sfu.ca (A.P. Punnen).

1572-5286/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.disopt.2010.08.001

consider an instance of a Hamiltonian path problem from a specified node s to a specified node t in a directed graph $D = (V, A)$. Without loss of generality, let us assume that D does not contain any arc incident into s, any arc incident out of t and arc (s, t). Construct an undirected graph $G' = (V', E')$ as follows: $V' = \{s, t\} \cup \{u', u'' : u \in V \setminus \{s, t\}\}$; $E' = \{(u', u'') : u \in V\} \cup \{e'_i : e_i \in A\}$, where for each $e'_i \in E'$, if $e_i = (u, v)$ then $e'_i = (s, v')$ if $u = s$, $e'_i = (u'', t)$ if $v = t$ and $e'_i = (u'', v')$ otherwise. Assign to each edge $e'_i \in E'$ cost c_{e_i}; and to each of the edges $\{(u', u'') : u \in V\}$ assign a zero cost. For every pair e'_i, e'_j of distinct edges in G' such that e_i and e_j are both incident into or both incident out of a common node u, let $\{e'_i, e'_j\} \in S$. Then it is easy to verify that MSTC problem on (G, S) is equivalent to the instance of Hamiltonian path problem. Another application of MSTC appears in [6,7] in their study of the *quadratic bottleneck spanning problem* (QBSTP) which is defined as follows. For each $(e, f) \in E \times E$ let w_{ef} be a prescribed weight. Then the QBSTP is to find a spanning tree T of G such that $\max\{w_{ef} : e, f \in T\}$ is as small as possible. Zhang and Punnen [6,7] consider the feasibility version of MSTC to develop exact and heuristic algorithms for the *quadratic bottleneck spanning tree problem* and it is the primary motivation for our work on MSTC. The optimization version of MSTC can be used to enhance heuristic algorithms for QBSTP to diverse search paths.

Let $\hat{G} = (\hat{V}, \hat{E})$ be the undirected graph with vertex set $\hat{V} = E$ and edge set $\hat{E}$ defined such that $(e, f) \in \hat{E}$ if and only if $\{e, f\} \in S$. Then $\hat{G}$ is called the *conflict graph*. Recently Darmann et al. [4], (See also Darmann et al. [5]), showed that MSTC is solvable in polynomial time if the associated conflict graph is a collection of disjoint edges, whereas the problem is NP-hard if the conflict graph is a collection of disjoint 2-edge paths with 0–1 edge costs. From this, it follows that computing a feasible solution or an ϵ-optimal solution to MSTC is NP-hard for any $\epsilon > 0$.

In this paper, we show that when G is a cactus, the feasibility version of the problem is polynomially solvable regardless of the structure of the conflict graph. This is somewhat surprising since MSTC is strongly NP-hard even when the conflict graph is a collection of disjoint 2-paths. Further, we show that the optimization version of MSTC is still NP-hard even on a cactus. MSTC can be solved in polynomial time if the conflict graph is a collection of disjoint cliques. This is achieved by showing that this special instance of MSTC reduces to a 2-matroid intersection problem [9,10]. Exploiting this result, we derive strong lower bounds for MSTC. To obtain one of these lower bounds we need to solve (exact or approximate) a *maximum edge clique partitioning problem (Max-ECP)* [11]. Max-ECP itself is NP-hard and it is known that the problem does not admit an $n^{1-O(1/(\log n)^{\gamma})}$ approximation, unless $NP \subseteq ZPTIME(2^{(\log n)^{O(1)}})$ for any fixed γ [12]. We show that Max-ECP can be solved by a polynomial time approximation algorithm with performance ratio $O(\frac{n(\log\log n)^2}{\log^3 n})$, improving the best known performance ratio of $O(n)$ for this problem [11]. Exploiting the relationship between Max-ECP, the independent set problem, and MSTC, several sufficient conditions are developed which, if satisfied, the problem can be declared to be infeasible. Further we observe that MSTC is polynomially solvable when the conflict graph becomes a collection of disjoint cliques after removing a fixed number of nodes. We then introduce heuristic algorithms to compute good quality approximate solutions. Preliminary computational results are reported.

The paper is organized as follows. In Section 2, we discuss various complexity results and polynomially solvable cases of MSTC. Section 3 deals with mathematical programming formulations of the problem and efficient lower bounding schemes. In Section 4 we discuss heuristics based on construction schemes, Lagrangian relaxation, local search, tabu search and tabu thresholding. Section 5 deals with sufficient conditions for infeasibility of MSTC. Computational results illustrating the efficacy of our heuristic algorithms, lower bound schemes, and infeasibility tests are presented in Section 6. Concluding remarks are given in Section 7.

For convenience we sometimes use the notations $V(G)$ and $E(G)$ to denote, respectively, the vertex set and edge set of a graph G.

2. Complexity and polynomially solvable cases

The complexity result of Darmann et al. [4] considers a general graph G with a very simple configuration for the conflict graph (disjoint 2-paths). This raises an interesting question: what is the complexity of MSTC when G has a simple configuration whereas $\hat{G}$ is arbitrary. Perhaps the simplest non-trivial candidate for G is a cactus. We now show that when G is a cactus (i.e. every edge in E lies on at most one cycle in G), the feasibility version of MSTC is solvable in polynomial time whereas the optimization version remains NP-hard. Suppose $G = (V, E)$ is a cactus, but the conflict set S is arbitrary. We assume, without loss of generality, that every edge in E lies on a cycle. This gives us a partition $(E_1, E_2, \ldots, E_k)$ of E where each $E_i = \{e_{1,i}, e_{2,i}, \ldots, e_{l_i,i}\}$ is the edge set of a cycle in G. Obviously, $l_i \geq 3\ \forall i$, and any $T \subseteq E$ is the edge set of a tree in G if and only if $|T \cap E_i| = l_i - 1\ \forall i$. Our problem thus reduces to choosing a set $T^* \subseteq E$ such that (i) $E - T^* = X^*$ contains precisely one edge from each E_i and X^* contains at least one element of each conflict pair in S, (feasibility), and (ii) $\sum_{e \in T^*} c_e$ is minimum, (or equivalently, $\sum_{e \in X^*} c_e$ is maximum), (optimality).

First of all, we shall show that we can assume, without loss of generality, that for each $i = 1, 2, \ldots, k$ the set S contains at most one 2-element set of the type $\{e_{p,i}, e_{q,i}\}$. For this, the following two operations will be useful.

Inclusion of an edge $e_{p,i}$ involves fixing $e_{p,i} \in X^*$ and deleting it from the set E_i, deleting from S all the sets of the type $\{e_{p,i}, e_{q,j}\}$ and performing the *exclusion* operation on all the edges in $E_i - \{e_{p,i}\}$, where the operation *exclusion* is defined as follows.

Exclusion of an edge $e_{p,i}$ involves deleting $e_{p,i}$ from the set E_i and performing the operation *inclusion* on all the edges $\{e_{q,j} : \{e_{p,i}, e_{q,j}\} \in S\}$.

Suppose for some $i \in \{1, 2, \ldots, k\}$, there exist more than one set of the type $\{e_{p,i}, e_{q,i}\}$ in S. If the intersection of all such sets is empty the problem is obviously infeasible; else if $e_{p,i}$ lies in all such sets then inclusion of $e_{p,i}$ gives us a reduced, equivalent problem.

We shall now show that the MSTC problem on cactus is equivalent to the well-known weighted 2-SAT problem [13]. From the results on the 2-SAT problem it will then follow that the feasibility version of our problem can be solved in $O(\max\{|E|, |S|\})$ time [14], while the optimality version of the problem is NP-hard [13]. To show the equivalence, we shall need the following problem.

Generalized 2-SAT problem (G2SAT): Here we are given a graph $\tilde{G} = (\tilde{N}, F)$ and a partition $(\tilde{N}_1, \tilde{N}_2, \ldots, \tilde{N}_k)$ of the set $\tilde{N}$, where $\tilde{N}_i = \{\tilde{n}_{1,i}, \tilde{n}_{2,i}, \ldots, \tilde{n}_{l_i,i}\}$ for some $l_i \geq 3$. For each node $p \in \tilde{N}$, a non-negative weight w_p is prescribed. The problem is to choose $\tilde{N}^* \subseteq \tilde{N}$ such that (i) $\tilde{N}^*$ contains exactly one node from each $\tilde{N}_i$ and each edge in F is incident to at least one node in $\tilde{N}^*$, and (ii) $\sum_{p \in \tilde{N}^*} w_p$ is maximum.

An instance of MSTC on a cactus can be formulated as G2SAT problem by defining $\tilde{n}_{i,j} = e_{i,j}$ and $w_{\tilde{n}_{i,j}} = c_{e_{i,j}}$, $\forall i, j$, and $\{\tilde{n}_{p,i}, \tilde{n}_{q,j}\} \in F$ if and only if $\{e_{p,i}, e_{q,j}\} \in S$.

Conversely, given an instance of G2SAT, define graph $G = (V, E)$ as follows: $V = \{0\} \cup \{i_j : i \in \{1, 2, \ldots, k\}, j \in \{i, 2, \ldots, l_i - 1\}\}$. Associate with each $\tilde{N}_i$ a cycle $(0, i_1, i_2, \ldots, i_{l_i-1}, 0)$, label the edges in this cycle as $E_i = \{e_{1,i}, e_{2,i}, \ldots, e_{l_i,i}\}$, and match each edge $e_{p,i}$ with element $\tilde{n}_{p,i}$ of $\tilde{N}_i$. Let $c_{e_{p,i}} = w_{\tilde{n}_{p,i}}$ $\forall p, i$ and let $\{e_{p,i}, e_{q,j}\} \in S$ if and only if $\{\tilde{n}_{p,i}, \tilde{n}_{q,j}\} \in F$. It is easy to see that $G = (V, E)$ is a cactus and this instance of MSTC is equivalent to G2SAT.

We now show that G2SAT is equivalent to the well-studied weighted 2-SAT problem [13], which can be stated as follows: We are given a collection $\{x_1, x_2, \ldots, x_k\}$ of k boolean variables each taking value either 1(=true) or 0(=false), a weight w_i for each x_i and a set $\{C_1, C_2, \ldots, C_m\}$ of m clauses, each of which is a 2-element subset of $\{v_1, v_2, \ldots, v_{2k}\} = \{x_1, \bar{x}_1, x_2, \bar{x}_2, \ldots, x_k, \bar{x}_k\}$, (here $\bar{x}_i = 1 - x_i$). The problem is to assign values to the variables such that (i) for each clause $C_i = \{v_p, v_q\}$, $v_p + v_q \geq 1$, and (ii) $\sum w_i x_i$ is maximum.

For a given instance of weighted 2-SAT problem, we construct an equivalent instance of G2SAT as follows: For each $i = 1, 2, \ldots, k$, define $\tilde{N}_i = \{\tilde{n}_{1,i}, \tilde{n}_{2,i}, \tilde{n}_{3,i}\}$, where $\tilde{n}_{1,i}$ corresponds to x_i, $\tilde{n}_{2,i}$ corresponds to $\bar{x}_i$ and $\tilde{n}_{3,i}$ is a dummy node. Let $w_{\tilde{n}_{1,i}} = w_i$, $w_{\tilde{n}_{2,i}} = w_{\tilde{n}_{3,i}} = 0$.

For any clause $C_i = \{v_p, v_q\}$, let $v_p \in \{x_i, \bar{x}_i\}$ and $v_q \in \{x_j, \bar{x}_j\}$. Add edge $(\tilde{n}_{a,i}, \tilde{n}_{b,j})$ to F, where $a = 1$ if $v_p = x_i$, $a = 2$ if $v_p = \bar{x}_i$, $b = 1$ if $v_q = x_j$ and $b = 2$ if $v_q = \bar{x}_j$. In addition, for each $i = 1, 2, \ldots, k$, add edge $(\tilde{n}_{1,i}, \tilde{n}_{2,i})$ to F.

It is easy to see that no solution $\tilde{N}^*$ to G2SAT contains a node $\tilde{n}_{3,i}$ for any i. From any solution to the weighted 2-SAT problem, we can construct a solution $\tilde{N}^*$ to G2SAT that has the same objective function value, and vice versa, as follows: For each $i = 1, \ldots, k$, $x_i = 1$ if and only if $\tilde{n}_{1,i} \in \tilde{N}^*$; $x_i = 0$ if and only if $\tilde{n}_{2,i} \in \tilde{N}^*$.

Now, suppose we are given an instance of G2SAT problem. We construct a corresponding instance of weighted 2-SAT problem as follows: Our variables are $\{x_{ij} : j = 1, 2, \ldots, k, \ i = 1, 2, \ldots, l_j\}$. With each edge $(\tilde{n}_{p,i}, \tilde{n}_{q,j}) \in F$, we associate a clause $C_{piqj} = \{x_{pi}, x_{qj}\}$. In addition, we create clauses $\{\{\bar{x}_{pi}, \bar{x}_{qi}\} : i = 1, 2, \ldots, k; \{p, q\} \subseteq \{1, 2, \ldots, l_i\}, p \neq q\}$. Let $w_{ij} = w_{\tilde{n}_{j,i}} + M$, where M is a sufficiently large integer.

It is easy to see that for any feasible solution to the instance of weighted 2-SAT problem, if for some $j \in \{1, 2, \ldots, k\}$, $x_{ij} = 0$ $\forall i$, then by arbitrarily choosing an $i \in \{1, 2, \ldots, l_j\}$ and making $x_{ij} = 1$, $\bar{x}_{ij} = 0$ gives us an alternate feasible solution. Using this, equivalence of sets of optimal solutions of the two problems as well as equivalence of the corresponding feasibility problems are easy to verify and the proof is omitted.

Since the weighted 2-SAT problem is NP-hard [13] but its feasibility version can be solved in $O(m)$ [14], we get the following theorem:

Theorem 1. *MSTC problem on a cactus is NP-hard. However, its feasibility version can be solved in* $O(\max\{|E|, |S|\})$ *time.*

We say that the conflict set (conflict relation) S is *transitive* if and only if for distinct e, f, g in E, $\{e, f\} \in S$, nad $\{f, g\} \in S$, imply $\{e, g\} \in S$.

Lemma 2. *The conflict graph* $\hat{G}$ *is a collection of disjoint cliques if and only if the conflict set* S *is transitive.*

The straightforward proof of Lemma 4 is omitted here. We now show that MSTC is polynomially solvable whenever S is transitive. Let $\hat{G}_1, \hat{G}_2, \ldots, \hat{G}_p$ be the connected components of the conflict graph $\hat{G} = (\hat{V}, \hat{E})$. Then by Lemma 2, each $\hat{G}_i$ is a clique. Let $V(\hat{G}_i) = E_i$ for $i = 1, 2, \ldots, p$. Then any conflict free tree T of G contains at most one edge from each E_i. Let F_1 be the family of all subsets of E satisfying the property that $Q \in F_1 \Leftrightarrow |Q \cap E_i| \leq 1$ for all i. Then (E, F_1) is a partition matroid [15]. Let F_2 be the collection of edge sets of all spanning trees of G. Then (E, F_2) is the base system of a graphic matroid [15]. Thus, in this case, MSTC reduces to the problem of finding a minimum cost set T in $F_1 \cap F_2$, i.e. MSTC is a special case of the well-solved weighted matroid intersection problem [10]. The preceding discussion is summarized in the theorem below.

Theorem 3. *When the conflict graph is a collection of disjoint cliques (equivalently when* S *is transitive), MSTC can be solved in polynomial time.*

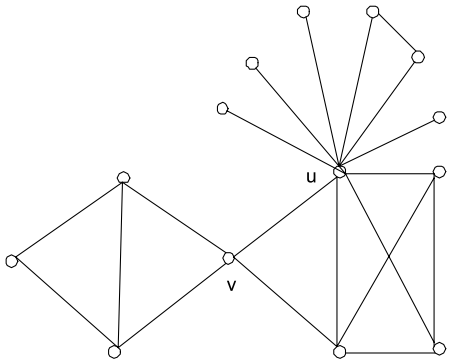

Fig. 1. Example of a graph satisfying conditions of Theorem 4. Here, $k = 2$ and nodes u and v are the nodes to be deleted.

Although the general weighted matroid intersection problem is polynomially solvable, it may be noted that special algorithms are available when the underlying matroids are of graphic and partition types with better worst case complexity [16]. Since our matroid intersection problem is precisely of this type, we can solve the problem in $O(nK^2 + nm + Kn^2)$ time, where K is the number of cliques. Our next two theorems indicate that when the conflict graph is slightly deviated from the structure of a collection of disjoint cliques, MSTC can still be solved in polynomial time.

Theorem 4. *Suppose the conflict graph $\hat{G}$ is such that deletion of a fixed number, k, of nodes, that can be identified in polynomial time, reduces $\hat{G}$ to a collection of node-disjoint cliques. Then the corresponding instance of MSTC can be solved in polynomial time.*

Proof. Consider each of the k nodes in $\hat{G}$, deletion of which reduces $\hat{G}$ to a collection of node-disjoint cliques. If the edge in G corresponding to such a node, x, is to be excluded from a tree, then we delete the edge from G and the node x from $\hat{G}$. If this edge is to be included in a tree, then we change its cost in G to $-M$, where M is a suitably large number and in $\hat{G}$, we delete all the neighbors of x from $\hat{G}$. There are 2^k ways one can choose the k nodes to be included or excluded. It can be verified that in any given such selection, after performing the above mentioned operations, the resulting conflict graph will be a collection of disjoint cliques. By Theorem 3, this problem can be solved in polynomial time. If the optimal solution does not include all the edges of cost $-M$, the selection is not feasible and we discard it. Otherwise, we compute the objective function value of the resulting tree with respect to the original costs. Repeating this for all possible 2^k selections and choosing the overall best tree gives us an optimal solution to the original MSTC. Since k is fixed, the result follows from Theorem 3. □

Note that in Theorem 4, although we fix the number of nodes to be deleted, the number of conflict pairs associated with these nodes need not be fixed since each such node can have a degree as large as $(n-1)$. Fig. 1 gives an example of a conflict graph satisfying conditions of Theorem 4.

As a corollary of Theorem 4, we have the following.

Corollary 5. *If the conflict graph has a fixed number k of edges, which can be identified in polynomial time, so that deletion of these edges makes the conflict graph a collection of disjoint cliques, then MSTC can be solved in polynomial time.*

When the conflict graph is not a collection of disjoint cliques (and also does not satisfy the condition of Theorem 4 for small value of k), one way to obtain a lower bound for MSTC is to relax some conflict relations (equivalently, delete some edges from $\hat{G}$) so that the resulting graph is a collection of disjoint cliques with maximum number of edges. This is precisely the maximum edge clique partitioning problem (Max-ECP) [11,17,12]. It is well known that Max-ECP is NP-hard and cannot be approximated within a factor of $n^{1-O(1/(\log n)^{\gamma})}$ unless $NP \subseteq ZPTIME(2^{(\log n)^{O(1)}})$. Although we are going to solve Max-ECP on the conflict graph, to keep the generality of our results, we assume that Max-ECP is defined on a general graph G for the rest of this section.

Dessmark et al. [11] showed that a maximum cardinality matching in G provides an ϱ-optimal solution to Max-ECP where ϱ is the largest cardinality of a clique in G. This is the best known performance ratio for a polynomial time approximation algorithm for the problem. Since ϱ can be $O(n)$, the best known data independent performance ratio for Max-ECP is $O(n)$. The bound ϱ can be slightly improved, (in a data dependent way), as follows by modifying the proof of performance ratio $O(\varrho)$ given in [11] and exploiting properties of possible alternative optimal solutions.

Let $\{\{V_1^k, V_2^k, \ldots, V_{m_k}^k\},\ k = 1, 2, \ldots, p\}$ be the set of all the optimal solutions to Max-ECP. For each k, define $\delta^k = \max_{1 \le j \le m_k} |V_j^k|$. Let $\delta = \min_{1 \le k \le p} \delta^k$.

Lemma 6. *If $M \subseteq E$ is a maximum cardinality matching in G and OPT is the optimum objective function value of Max-ECP on G then $|M| \ge \frac{OPT}{\delta}$.*

Proof. Let $\{V_1, V_2, \ldots, V_t\}$ be an optimal solution to Max-ECP such that $\max_{1\le i\le t} |V_i| = \delta$. Let $M^0 = \cup_{i=1}^{t} M_i^0$, where M_i^0 is a maximum cardinality matching in the subgraph of G induced by V_i. Then

$$\begin{aligned}
|M| \ge |M^0| &= \sum_{i=1}^{t} |M_i^0| \\
&\ge \sum_{i=1}^{t} \frac{|V_i| - 1}{2} \\
&\ge \sum_{i=1}^{t} \frac{|V_i|}{\delta} \left(\frac{|V_i| - 1}{2} \right) \\
&\ge \frac{1}{\delta} \sum_{i=1}^{t} |V_i| \left(\frac{|V_i| - 1}{2} \right) = \frac{1}{\delta} OPT. \quad \square
\end{aligned}$$

Note that $\delta \le \varrho$. As in the case of ϱ, the ratio δ could also be $O(n)$. However, the advantage of Lemma 6 is that it links the performance ratio to the smallest value of the size of the largest clique in an optimal clique partition. Let us now discuss how to improve this bound. We consider a variation of the greedy algorithm [11], called *the approximate greedy algorithm* that iteratively extracts a clique of reasonably large size from the conflict graph G. A formal description of the algorithm is given below. We assume that a procedure Approx-Clique(G) is available which with input a graph G outputs an approximate solution (clique) for the maximum clique problem on G.

Algorithm 1: The Approximate Greedy Algorithm

Input: The graph G;
$H = \emptyset$, $G^1 = G$, $k = 1$;
while $E(G^k) \ne \emptyset$ **do**
  D^k = Approx-Clique(G^k);
  $H = H \cup \{D^k\}$;
  $G^{k+1} = G^k - V(D^k)$, $k = k + 1$;
end while;
Output: $H \cup \{G^k\}$.

Theorem 7. *If Approx-Clique(G) computes an ϵ-optimal solution to the maximum clique problem on G, then the approximate greedy algorithm computes a $(3\epsilon - 1)$-optimal solution to Max-ECP on G. Further if the complexity of Approx-Clique(G) is $O(f(m, n))$ then the complexity of the approximate greedy algorithm is $O(nf(m, n))$.*

Proof. In each iteration the algorithm considers a subgraph G^k of G and extracts an ϵ-optimal solution D^k for the maximum clique problem on G^k. The algorithm terminates when G^k becomes empty or a collection of isolated nodes. Since $|V(D^k)| \ge 2$ for all k, (except possibly the last) the total number of iterations is $O(n)$ and hence the complexity result follows. Let us now analyze the performance ratio. Let $Q^1 = (Q_1^1, Q_2^1, \ldots, Q_t^1)$ be an optimal solution to Max-ECP on G. Let ϱ^k be the size of the maximum clique in G^k. Since D^k is an ϵ-optimal solution to the maximum clique problem on G^k, we have

$$\varrho^k \le \epsilon |V(D^k)|. \tag{1}$$

Let $|V(D^k)| = d^k$ for all k. Then $|E(D^k)| = \frac{d^k(d^k-1)}{2}$. Note that

$$G^{k+1} = G^k - V(D^k) \quad \text{for } k = 1, 2, \ldots, (r-1)$$

where r is the number of iterations in the algorithm. Let $Q_i^{k+1} = Q_i^k - V(D^k)$. Then $Q^{k+1} = (Q_1^{k+1}, Q_2^{k+1}, \ldots, Q_t^{k+1})$ is a clique partition of G^{k+1}. Now $|Q_i^k| \le \varrho^k \le \epsilon d^k$ for all $i = 1, 2, \ldots, t$. Thus $|E(Q_i^k)| - |E(Q_i^{k+1})| \le |Q_i^k \cap D^k|(\epsilon d^k - 1)$ if $Q_i^k \cap D^k \ne \emptyset$ and $|E(Q_i^k)| - |E(Q_i^{k+1})| = 0$ if $Q_i^k \cap D^k = \emptyset$. Note that $\sum_{i=1}^{t} |Q_i^k \cap D^k| = d^k$. Thus

$$\sum_{i=1}^{t} |E(Q_i^k)| - |E(Q_i^{k+1})| \le d^k(\epsilon d^k - 1). \tag{2}$$

Adding (2) for $k = 1, 2, \ldots, r$ and noting that $|E(Q_i^{r+1})| = 0$ we have

$$\begin{aligned}
\sum_{i=1}^{t} |E(Q_i^1)| &\le \sum_{k=1}^{r} d^k(\epsilon d^k - 1) \\
&= \sum_{k=1}^{r} (d^k \epsilon (d^k - 1) + \epsilon - 1)
\end{aligned}$$

$$= 2\epsilon \sum_{k=1}^{r} \frac{d^k(d^k-1)}{2} + r(\epsilon - 1)$$
$$= 2\epsilon |E(H)| + r(\epsilon - 1)$$

where H is the output of the algorithm. Since $r \leq |E(H)|$ we have,

$$\frac{|E(Q^1)|}{|E(H)|} \leq 2\epsilon + \frac{r(\epsilon-1)}{|E(H)|} \leq 2\epsilon + \epsilon - 1 = 3\epsilon - 1.$$

The result now follows from the optimality of Q^1 to Max-ECP on G. □

In the approximate greedy algorithm, if we replace Approx-Clique (G) by an exact optimization algorithm for the maximum clique problem on G (i.e. $\epsilon = 1$) we get a performance ratio of 2. In this case our algorithm reduces to the greedy algorithm of Dessmark et al. [11] and hence Theorem 7 is a proper generalization of the corresponding result of [11]. Since the maximum clique problem can be approximated within a factor of $\frac{n(\log\log n)^2}{(\log n)^3}$ [18] we have

Corollary 8. *The problem Max-ECP has a polynomial time ϵ-approximation algorithm where $\epsilon = (\frac{3n(\log\log n)^2}{(\log n)^3} - 1)$.*

The previously best known performance ratio for a polynomial time approximation algorithm for Max-ECP is $O(n)$ [11] and Corollary 8 improves this bound.

3. Integer programming formulations and lower bounds

Let $E = \{1, 2, \ldots, m\}$ and T be the edge set of a spanning tree of G. The incidence vector $x = (x_1, x_2, \ldots, x_m)$ of T is defined as

$$x_i = \begin{cases} 1 & \text{if } i \in T \\ 0 & \text{Otherwise.} \end{cases}$$

Let $\mathcal{F}$ be the spanning tree polytope of G. i.e. $\mathcal{F}$ is the convex hull of incidence vectors of spanning trees of G. Then MSTC can be formulated as a 0–1 integer linear program

ILP: $\min \sum_{e\in E} c_e x_e$

Subject to

$$\begin{aligned} & x \in \mathcal{F} \\ & x_e + x_f \leq 1 \quad \forall \{e, f\} \in S \\ & x_e = 0 \text{ or } 1 \quad \forall e \in E. \end{aligned} \tag{3}$$

Another integer programming formulation of MSTC can be described as follows:

ILP-Star: $\min \sum_{e\in E} c_e x_e$

Subject to

$$\begin{aligned} & x \in \mathcal{F} \\ & d_e x_e + \sum_{j \in \hat{V}(e)} x_j \leq d_e \quad \forall e \in E \\ & x_e = 0 \text{ or } 1 \quad \forall e \in E \end{aligned} \tag{4}$$

where $\hat{V}(e)$ is the set of nodes in $\hat{G}$ that are adjacent to e and $d_e = |\hat{V}(e)|$. It can be verified that the set of binary solutions of (3) and (4) are equivalent and hence ILP and ILP-star are equivalent. We call constraints (4) the *star inequalities*. ILP-Star gives a more compact representation of MSTC than ILP.

MSTC can also be formulated as a *quadratic minimum spanning tree problem* (QMSTP) as follows:

$$\min \sum_{e\in E} \sum_{f\in E} d_{ef} x_e x_f$$

Subject to

$$\begin{aligned} & x \in \mathcal{F} \\ & x_e = 0 \text{ or } 1 \quad \forall e \in E \end{aligned}$$

where $D = (d_{ef})$ is the $m \times m$ matrix defined as

$$d_{ef} = \begin{cases} c_e & \text{if } e = f \\ M & \text{if } \{e, f\} \in S, e \neq f \\ 0 & \text{if } \{e, f\} \notin S, e \neq f \end{cases}$$

and M is a large number.

The QMSTP has been studied by various authors [19–22] who proposed exact and heuristic algorithms to solve the problem. These algorithms can be used to solve MSTC as well. However, exploiting the special structure of MSTC, we develop additional heuristic algorithms.

Let us now focus our attention to computing good quality lower bounds for the problem MSTC. We first consider a somewhat straightforward lower bound using the Lagrangian relaxation of constraints (3) in ILP. Let λ_{ef} be the Lagrangian multiplier associated with conflict constraint $x_e + x_f \leq 1$ in ILP for all $\{e, f\} \in S$. For any node e in the conflict graph $\hat{G}$, let $\hat{V}(e)$ be the set of its adjacent nodes. Consider the Lagrangian function

$$\begin{aligned} L(\lambda) &= \min \left\{ \sum_{e \in E} c_e x_e + \sum_{\{e,f\} \in S} \lambda_{ef}(x_e + x_f - 1) : x \in \mathcal{F} \right\} \\ &= -\sum_{\{e,f\} \in S} \lambda_{ef} + \min \left\{ \sum_{e \in E} \left(c_e + \sum_{f \in \hat{V}(e)} \lambda_{ef} \right) x_e : x \in \mathcal{F} \right\}. \end{aligned}$$

From the Lagrangian duality, $L(\lambda)$ is a lower bound for the optimal objective function value of MSTC for each $\lambda \geq 0$. Thus

$$L^* = \max_{\lambda \geq 0} L(\lambda)$$

is a lower bound on the optimal objective function value of MSTC. Note that L^* can be obtained using any algorithm for a non-differentiable convex optimization problem; in particular the subgradient algorithm [23] or the volume algorithm [24]. In each subgradient iteration, we need to evaluate $L(\lambda)$ for a given vector $\lambda = (\lambda_{ef} : \{e, f\} \in S)$. This can be accomplished by solving the minimum spanning tree problem:

$$\begin{aligned} &\min \left\{ \sum_{e \in E} \left(c_e + \sum_{f \in \hat{V}(e)} \lambda_{ef} \right) x_e \right\} \\ &\text{Subject to} \\ &\qquad x \in \mathcal{F}. \end{aligned}$$

It is possible to obtain a lower bound, say L^*_{Star} similar to the one discussed above, by considering the Lagrangian relaxation of the star inequalities of ILP-Star. However, it is observed that the resulting bound is inferior to the bound L^* and the computational advantage in this case is not significant. Thus we discarded L^*_{Star} from further investigation.

3.1. Improving the lower bound

The lower bound L^* obtained above can be improved by investing additional computational effort. Let Δ be a subset of the edge set of conflict graph $\hat{G}$ such that the graph $\tilde{G} = \hat{G} - \Delta$ is a collection of disjoint cliques. From Theorem 3, MSTC on G with $\tilde{G}$ as conflict graph can be solved in polynomial time. To exploit this information in computing an improved lower bound, we rewrite ILP as

$$\begin{aligned} \text{ILP-MI: } &\min \sum_{e \in E} c_e x_e \\ &\text{Subject to} \\ &\qquad x \in \mathcal{F} \\ &\qquad x_e + x_f \leq 1 \quad \forall (e, f) \in E(\tilde{G}) && (5) \\ &\qquad x_e + x_f \leq 1 \quad \forall (e, f) \in \Delta && (6) \\ &\qquad x_e = 0 \text{ or } 1 \quad \text{for all } e. && (7) \end{aligned}$$

Let $\bar{G}$ be the subgraph of $\hat{G}$ with edge set Δ. For any node e in the graph $\bar{G}$, let $\bar{V}(e)$ be the set of its adjacent nodes. Consider the Lagrangian function $\ell(\lambda)$ obtained by relaxing (6) in ILP-MI. We have,

$$\begin{aligned} \ell(\lambda) &= \min \left\{ \sum_{e \in E} c_e x_e + \sum_{(e,f) \in \Delta} \lambda_{ef}(x_e + x_f - 1) : x \in \mathcal{F}, x_e + x_f \leq 1 \, \forall (e, f) \in E(\tilde{G}) \right\} \\ &= -\sum_{(e,f) \in \Delta} \lambda_{ef} + \min \left\{ \sum_{e \in E} \left(c_e + \sum_{f \in \bar{V}(e)} \lambda_{ef} \right) x_e : x \in \mathcal{F}, x_e + x_f \leq 1 \, \forall (e, f) \in E(\tilde{G}) \right\}. \end{aligned}$$

From the Lagrangian duality,

$$\ell^* = \max_{\lambda \geq 0} \ell(\lambda)$$

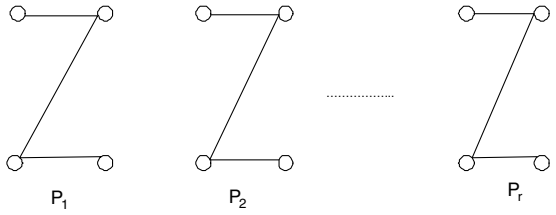

Fig. 2. Example: matching heuristic vs approximate greedy.

is a lower bound on the optimal objective function value of MSTC. As discussed in the case of L^*, the lower bound ℓ^* can be obtained using the subgradient algorithm [23] or using the volume algorithm [24]. In each subgradient iteration, we need to evaluate $\ell(\lambda)$ for a given vector $\lambda = (\lambda_{ef} : (e, f) \in \Delta)$. This can be accomplished by solving the weighted matroid intersection problem:

$$\text{MI}(\lambda):\ \min \sum_{e\in E}\left(c_e + \sum_{f\in\bar{V}(e)} \lambda_{ef}\right) x_e$$

Subject to

$$x \in \mathcal{F}$$
$$x_e + x_f \le 1 \quad \forall (e, f) \in E(\tilde{G}). \tag{8}$$

It is easy to see that $\ell^* \ge L^*$. The quality of the lower bound ℓ^* depends on clever choices of Δ. Ideally we want to choose Δ such that $|\Delta|$ is as small as possible so that the resulting graph $\tilde{G}$ is a collection of disjoint cliques with maximum number of edges. But the optimal choice of such a $\tilde{G}$ is a difficult problem since this selection problem is precisely the *maximum edge clique partitioning problem* (Max-ECP) [11,17,12], discussed earlier in this paper, applied on the graph $\hat{G}$. An approximate solution to Max-ECP can be identified using the approximate greedy algorithm discussed in Section 2.

Another heuristic algorithm to solve Max-ECP on $\hat{G}$ (i.e. to generate $\tilde{G}$ from $\hat{G}$) is obtained by designating $\tilde{G}$ as a maximum cardinality matching in $\hat{G}$. We call this the *matching heuristic*. Again, the performance ratio for matching heuristic was discussed in Section 2. Although the approximate greedy heuristic may appear stronger than the matching heuristic in general, it may be noted that the matching heuristic could perform significantly better in certain classes of graphs. For example consider the graph in Fig. 2 which consists of a collection of r disjoint 3-edge paths.

Potentially, the approximate greedy heuristic could output the central edge from each of the 3-edge paths resulting in a solution of cardinality r where as the matching heuristic will produce the optimal solution to Max-ECP with cardinality $2r$.

4. Upper bounds and heuristics

We now discuss several heuristics for the MSTC. These algorithms include construction heuristics, local search, tabu search, tabu thresholding, and the Lagrangian based heuristics. As noted earlier, computing a feasible solution to MSTC itself is NP-hard. Thus, our heuristic algorithms may not always compute a feasible solution. When a feasible solution is not obtained, we try to minimize the number of violated conflict constraints.

4.1. A simple construction heuristic

A very fast construction heuristic for MSTC can be obtained as follows. In the graph G we choose an $e \in E$ which appears in the maximum number of conflict pairs, delete e from G and delete from S all the conflict pairs containing e. We repeat this process with the updated G and S, until $S = \emptyset$. If G is still connected, then the cost of its minimum spanning tree is an upper bound to the MSTC, otherwise the algorithm returns no feasible solution.

4.2. Local search

Let T be a spanning tree and $e \in E - T$, i.e. e is a non-tree edge. The graph $T + e$ contains a unique cycle with edges $e_1, e_2, \ldots, e_{t_e}$. Then $T_i = T + e - e_i$ is a spanning tree of G, for $1 \le i \le t_e$. Let $R(e) = \{T + e - e_i : i = 1, 2, \ldots, t_e\}$ and $N(T) = \cup_{e\in E-T} R(e)$. The set $N(T)$ is called a *2-exchange neighborhood*. The neighborhood $N(T)$ is well known and is used by many researchers for developing local search algorithms for spanning tree type problems. We use this neighborhood to develop a local search heuristic for MSTC.

Since finding a feasible solution to MSTC is NP-hard, to have a meaningful local search algorithm we consider a modified objective function $f(T) = c(T) + \alpha g(T)$ where $c(T) = \sum_{e\in T} c_e$, α is a large number and $g(T) = |\{(e, f) : \{e, f\} \in S \text{ and } e, f \in E(T)\}|$. Thus $g(T)$ measures the number of violated conflict constraints. The spanning tree T is feasible if and only if $g(T) = 0$.

For any edge e of G let $Z(e, T) = \{h : \{e, h\} \in S, h \in T, e \neq h\}$. Given the value of $f(T)$ the objective function value of the tree $T_i = T + e - e_i$ can be computed as $f(T_i) = C(T_i) + \alpha g(T_i)$ where $C(T_i) = C(T) + c_e - c_{e_i}$ and

$$g(T_i) = \begin{cases} g(T) + |Z(e, T)| - |Z(e_i, T)| - 1 & \text{if } \{e, e_i\} \in S \\ g(T) + |Z(e, T)| - |Z(e_i, T)| & \text{otherwise.} \end{cases}$$

Note that the size of the neighborhood $N(T)$ is $O(mn)$ and the neighborhood can be searched for an improving solution in $O(mn)$ time by maintaining appropriate data structure.

The local search algorithm for MSTC is now straightforward. Starting with a spanning tree T, the algorithm examines the neighborhood $N(T)$ and move to the first improving solution, and then we check the neighborhood of the new solution for the first improving solution again. This recursion will stop when a local optimum T^* is reached. If $g(T^*) = 0$, the local optimum solution is feasible. Otherwise, we report the objective function values in terms of $C(T)$ and $g(T)$ where $g(T)$ gives the number violated conflict constraints.

4.3. Tabu thresholding heuristic

Whether the local search terminates with a feasible solution or not, further improvements in terms of reducing the number of conflict pair violations or improving the objective function value of a feasible solution can be explored by careful and systematic search procedures. Tabu thresholding [25] is one such strategy that can be used to achieve this goal. This approach was useful in finding good quality solutions for various combinatorial optimization problems [26,27]. Tabu thresholding can be viewed as a special randomized local search algorithm [28]. The algorithm uses two parameters δ and ω to control solution quality and search diversification.

Suppose T is a local solution as defined in the local search algorithm. Let $T_1, T_2, \ldots, T_\sigma$ be the spanning trees of G in $N(T)$ where $f(T_1) \leq f(T_2) \leq \cdots \leq f(T_\sigma)$. Choose a spanning tree T^0 randomly from $\{T_1, T_2, \ldots, T_\omega\}$ where ω is a parameter. Move to the solution T^0 and we call such a move *reasonably random move*. Continue the reasonably random moves for δ iterations, where δ is another parameter that controls the algorithm. After completing the reasonably random move phase, we switch back to local search phase until a local minimum is reached and then reasonably random moves are invoked. We continue the process until a prescribed number, say *max-iter*, of iterations (number of times local search is invoked) are completed and we output the best solution obtained.

4.4. Tabu search heuristic

Another way to explore the neighborhood $N(T)$ beyond a local optimum is to employ the tabu search method [29,30, 20]. For the current solution T, if for some $e_i \in E \setminus T$, $e_j \in T$, $T + e_i - e_j$ is a spanning tree of G, we call (e_i, e_j) a *2-exchange pair*. The tabu list we construct in this algorithm consists of some 2-exchange pairs. We define $N_A(T) = \cup_{e_i \in E \setminus T}\{T + e_i - e_j : (e_i, e_j)$ *is 2-exchange pair*, $(e_i, e_j) \notin$ *Tabu List*$\}$, so $N_A(T) \subseteq N(T)$ and for any moves in $N_A(T)$, the corresponding 2-exchange pairs are not on the tabu list.

In each iteration, we first find the best solution T^0 in $N(T)$. Suppose $T^0 = T + e_i^0 - e_j^0$, then if (e_i^0, e_j^0) is not on the tabu list, we will move from T to T^0 and add (e_i^0, e_j^0) to the tabu list. To keep the length of the tabu list within a fixed length L, we add the new pair to the end of the list and once the length of the list is L, the pair on the top will be removed. If (e_i^0, e_j^0) is on the tabu list, we will check the aspiration criteria, which is set as $f(T^0)$ is less than the best objective function value so far. If the aspiration criteria is satisfied, we will make the move without updating the tabu list, otherwise we will find the best solution $\tilde{T}^0$ in $N_A(T)$ and move to $\tilde{T}^0$. The process is continued until a prescribed number of iterations (*max-iter*) is completed and output the best solution obtained.

4.5. Lagrangian heuristic

When solving the Lagrangian problem of ILP or ILP-MI, using subgradient optimization, we generate a sequence of spanning trees. It is useful to keep track of the objective function value $f(T)$ of these solutions and the best solution so obtained is output as a heuristic solution.

5. Infeasibility tests

Note that testing if MSTC has a feasible solution or not is an NP-hard problem. We now develop some sufficiency conditions that can be used to test infeasibility of the problem.

Lemma 9. *Let α be an upper bound on the independence number of the conflict graph $\hat{G}$. If $\alpha < |V(G)| - 1$ then MSTC is infeasible.*

Proof. Note that nodes of $\hat{G}$ are precisely edges of G. Let κ be the independence number of $\hat{G}$. For any feasible tree T of G, $E(T)$ corresponds to an independent set of $\hat{G}$. Thus $|V(G)| - 1 = |E(T)| \leq \kappa$. Thus, if $|V(G)| - 1 > \kappa$, MSTC must be infeasible. If $|V(G)| - 1 > \alpha$ holds, then $|V(G)| - 1 > \kappa$ holds and hence MSTC is infeasible. □

Although computing κ in Lemma 9 is NP-hard, reasonable values of α can be obtained by solving various relaxations of the maximum independent set problem on $\hat{G}$. We used the following scheme to get an estimate of α. Consider the integer programming formulation of the maximum independent set problem and invoke an integer programming solver (we used CPLEX) using branch and bound/ branch and cut algorithm. At any stage of the algorithm, if the smallest upper bound among the upper bounds at active nodes is less than $n-1$, by Lemma 9 we can conclude that the MSTC is infeasible. Also, at any stage of the algorithm, if an independent set of size $n - 1$ or more is found, we can conclude that the test fails and the algorithm terminates. The algorithm can also be terminated based on computational time constraints. We call this infeasibility test, the *maximum independent set test* (MIS test).

Lemma 10. *If β is a lower bound on the size of the minimum vertex cover of $\hat{G}$ and $|E(G)| - \beta < |V(G)| - 1$ then MSTC is infeasible. Further, if $D_1, D_2, \ldots, D_r$ is a clique partitioning of $\hat{G}$ and $|V(G)| - 1 > |E(G)| + r - \sum_{i=1}^{r} |D_i|$ then MSTC is infeasible.*

Proof. Let $m = |E(G)| = |V(\hat{G})|$ and $n = |V(G)|$. If β is a lower bound on minimum vertex cover of $\hat{G}$, them $m - \beta$ is an upper bound on the independence number of $\hat{G}$. The first part of the result follows from Lemma 9. If $D_1, D_2, \ldots, D_r$ is a clique partitioning of $\hat{G}$, then any vertex cover must select at least $|D_i| - 1$ vertices from D_i for each i. Thus $\sum_{i=1}^{r} |D_i| - r$ is a lower bound for the size of minimum vertex cover of $\hat{G}$ and the result follows. □

Corollary 11. *Let M be a maximum cardinality matching in $\hat{G}$. If $|V(G)| - 1 > |E(G)| - |M|$ then MSTC is infeasible.*

Note that a clique partition of $\hat{G}$ is constructed for computing the lower SUB + MI. This can be exploited to apply an infeasibility test based on the second part of Lemma 10 and we call it the *clique partition test* (CP test). Also, similar to the MIS test, Lemma 10 can be used to develop another test based on the minimum vertex cover problem. Consider the integer programming formulation of the minimum vertex cover problem and apply an integer programming solver (we used CPLEX) using branch and bound/ branch and cut algorithm. At any stage of the algorithm, if the largest lower bound among the lower bounds at the active nodes is less than $n - 1$, by Lemma 10 we can conclude that the MSTC is infeasible. Also, at any stage of the algorithm, if a vertex cover of size $m - n + 1$ or more is found, we can conclude that the test fails and the algorithm terminates. The algorithm can also be terminated based on computational time constraints. We call this feasibility test, the *minimum vertex cover test* (MVC test).

ILP can be reformulated with a compact representation using the single commodity formulation of the minimum spanning tree problem [31] along with the conflict constraints. Let $\vec{G}$ be the directed version of G obtained by replacing each undirected edge $\{i, j\}$ of G by two directed edges (i, j) and (j, i). Let $\delta^+(i) = \{j : (i, j) \in V(\vec{G})\}$, $\delta^-(i) = \{j : (j, i) \in V(\vec{G})\}$ and $V(G) = \{1, 2, \ldots, n\} = V(\vec{G})$. The following formulation gives a compact integer programming representation of MSTC.

$$
\begin{aligned}
\text{CILP}\quad & \min \sum_{e \in E(G)} c_e x_e \\
& \text{Subject to} \\
& \sum_{j \in \delta^+(1)} g_{1j} - \sum_{j \in \delta^-(1)} g_{j1} = n - 1 \\
& \sum_{j \in \delta^-(i)} g_{ji} - \sum_{j \in \delta^+(i)} g_{ij} = 1 \quad \text{for all } i \in V(G) \setminus \{1\} \\
& g_{ij} \leq (n-1)x_e \quad \text{for every edge } e = \{i, j\} \\
& g_{ji} \leq (n-1)x_e \quad \text{for every edge } e = \{i, j\} \\
& \sum_{e \in E} x_e = n - 1 \\
& x_e + x_f \leq 1 \quad \forall (e, f) \in S \\
& g_{ij} \geq 0, g_{ji} \geq 0, x_e = 0 \text{ or } 1 \quad \text{for all } e = \{i, j\} \in E(G).
\end{aligned}
$$

Solving CILP using CPLEX with a time limit of 1000 s was used as yet another feasibility test. By increasing the permitted running time, we used CILP to explore exact optimal solutions also.

6. Computational results

The experiments were conducted primarily on two machines: on a Dell PC with 3.40 GHz Intel pentium processor and 2.0 GB memory running Windows XP operation system and a Dell workstation with a 2.0 GHz Intel Xeon processor and 512 MB of memory running the Linux operating system. The lower bound algorithms, feasibility tests, and heuristics were

coded in C++. The public domain compiler Dev C++ was used for all compilation works on the PC and GNU gcc 3.6 compiler was used on the workstation. All general integer programs were solved using CPLEX 9.1 on the workstation. The goals of this preliminary experimental study were to (1) identify relative strengths of our lower bounds; (2) examine the strength of the lower bounds in relation to heuristic upper bounds; (3) examine the relative quality of the heuristic solutions and (4) assess the quality of the infeasibility tests.

There are no standard benchmark problems available for MSTC. Thus random test instances are generated as follows. First, a random connected graph G is constructed with the number of nodes ranging from 10 to 300 and the number of edges from 20 to 1000. Edge costs are random integers in the interval [0, 500]. Note that the conflict set determines if the problem is feasible or not. Given a random graph G, let S be the conflict set and $p = |S|$. If p is large, the corresponding MSTC is likely to be infeasible. Keeping this in mind, we have generated two types of conflict sets and the resulting problem instances are classified as type 1 and type 2. For type 1 problems, p is selected randomly as an integer in the range $[\lceil \frac{m}{100} \rceil, 15\lceil \frac{m}{100} \rceil]$ where m is the number of edges. Thus the number of conflict pairs in this class is between 1% and 15% of the total number of edges. There is no guarantee that these problems are feasible. For the type 2 class of problems, we make sure that the instances generated are feasible. For this, we first choose $p \in [25\lceil \frac{m}{100} \rceil, 35\lceil \frac{m}{100} \rceil]$ and generate a random conflict set S^*. The resulting MSTC instance is solved using local search with a random spanning tree as the starting solution and the objective function $g(T)$ is chosen to guide the search. If a feasible solution is obtained, then S is chosen as S^*. Otherwise, conflict pairs violated by the resulting solution are deleted from S^* and the remaining conflict pairs form the set S. Thus, the type 2 instances generated are guaranteed to be feasible.

Experiments are conducted using the following algorithms or combination of algorithms:

(1) Sub + MST: subgradient algorithm solving the Lagrangian relaxation problem of ILP to compute lower bound;
(2) Sub + MI: subgradient algorithm solving the Lagrangian relaxation problem of ILP-MI to compute lower bound; To obtain the conflict pairs such that the corresponding conflict graph is a collection of disjoint cliques, we use the heuristic algorithm [18] to compute a maximum clique within our greedy algorithm for Max-ECP.
(3) LS: local search algorithm using the Sub + MST solution as the starting solution;
(4) TT: tabu thresholding algorithm with the spanning tree obtained by solving

$$\begin{aligned} &\min \sum_{\{e,f\}\in S} (x_e + x_f) \\ &\text{Subject to} \\ &x \in \mathcal{F} \end{aligned} \tag{9}$$

as the starting solution. Default parameter values are set as Max-Iter $= 10$, $\delta = 10$, and $\omega = 15$.
(5) TS: tabu search algorithm with the same starting solution as that of tabu thresholding. Default parameter values are set as Tabu-size $= 7$ and Max-Iter $= 100$.
(6) Infeasibility sets: CP test, MIS test, MVS test, and CILP with a time limit.

Note that Sub + MST and Sub + MI compute lower bounds. We set the maximum number of subgradient iterations to 100, and the Lagrangian multipliers are updated using standard way of using small step lengths.

In the class of type 1 problems, 85 instances are generated using different values of the triplet (n, m, p) where n is the number of nodes, m is the number of arcs and p is the cardinality of S. From the experiments on these instances, possible outcomes are: (1) some problems are identified as infeasible, (2) some problems for which an ILP solver/heuristic was able to compute an optimal/feasible solution and (3) feasibility is not known for some problems. In the first set of experiments, we applied our infeasibility tests. This eliminated several type 1 problems from further consideration.

The MIS test solves the integer program

$$\begin{aligned} &\max \sum_{e\in V(\hat{G})} x_e \\ &\text{Subject to} \\ &x_e + x_f \le 1 \end{aligned}$$

using CPLEX with a time limit of 1000 s. Also, the algorithm is terminated when all active nodes have an upper bound value less than $n - 1$ with a certificate of infeasibility. At any stage, if the algorithm obtained an independent set of size $n - 1$ or larger, we terminate the algorithm with a flag that the MIS test failed. If none of these termination criterion is satisfied after 1000 s of CPU time was elapsed, the algorithm is terminated with the flag that MIS test failed. Instead of solving the integer program, we also experimented with its linear programming relaxation and the test was not very effective, but the integer programming version as explained above identified infeasibility of several problems. The MVC test solves the integer program

$$\begin{aligned} &\min \sum_{e\in V(\hat{G})} x_e \\ &\text{Subject to} \\ &x_e + x_f \ge 1 \end{aligned}$$

Table 1
Type 1 problems—Infeasible.

n	m	p	CP test	MIS test	MVC test	CPLEX
50	200	1990	–	–	722.68	618.22
50	200	2985	–	6.7	7.42	9.34
100	300	2242	–	64.73	67.85	65.09
100	300	4484	0.375	0.15	0.16	0.83
100	300	6726	0.484	0.22	0.22	0.64
200	400	798	–	–	–	0.12
200	400	1596	0.625	0.07	0.06	0.15
200	400	2394	0.593	0.08	0.09	0.15
200	400	3990	0.615	0.11	0.11	0.17
200	400	7980	0.735	0.26	0.26	0.25
200	400	11970	0.875	0.47	0.44	0.35
200	600	8985	–	1.61	1.63	6.25
200	600	17970	2.000	0.96	0.93	4.38
200	600	26955	2.375	1.82	1.87	4.06
200	800	31960	–	25.17	29.79	49.68
200	800	47940	5.297	12.49	10.33	56.92
300	600	1797	2.079	0.07	0.09	0.21
300	600	3594	1.803	0.09	0.13	0.25
300	600	5391	1.750	0.15	0.17	0.3
300	600	8985	1.687	0.25	0.29	0.37
300	600	17970	2.297	0.7	0.78	0.55
300	600	26955	2.291	1.39	1.58	0.87
300	800	6392	–	768.89	598.14	2.97
300	800	9588	4.406	0.56	0.62	0.45
300	800	15980	4.422	0.72	0.68	0.53
300	800	31960	5.109	1.82	1.87	0.87
300	800	47940	5.531	3.47	3.52	1.20
300	1000	24975	–	18.46	18.05	60.96
300	1000	49950	8.829	7.32	7.42	58.79
300	1000	74925	11.281	6.48	6.25	40.53

using CPLEX with a time limit of 1000 s. As in the case of MIS test, the algorithm is terminated when all active nodes have a lower bound value less than $m-n+1$ with a certificate of infeasibility. At any stage if the algorithm obtained a vertex cover of size $m-n+1$ or larger, we terminate the algorithm with a flag that the MVC test failed. If none of these termination criterion is satisfied after 1000 s of CPU time, the algorithm is terminated with the flag that MVC test failed. As in the case of MIS test, linear programming relaxation of this integer program was not very successful as a good bound to use in the test but the integer version, as discussed above, proved to be very effective.

Out of the 85 type 1 test problems generated, 30 turned out to be provably infeasible. These instances are tabulated in Table 1. In the table, a "-" indicates that the infeasibility test failed and the number in a column shows the CPU time for the corresponding test so that the problem is conclusively identified as infeasible. The column CPLEX in the table corresponds to solving CILP using the integer programming solver CPLEX with a time limit of 1000 s.

From the experimental results it can be seen that all infeasibility tests performed reasonably well. Infeasibility is detected in majority of cases in less than 50 s for most problems. In Table 2, we summarize experimental results on type 1 problems where a feasible solutions is obtained. For the two lower bound algorithms and heuristics, we recorded the bounds 'LB' and 'UB' along with CPU times. The column 'CPLEX' contains the optimal objective function value 'obj' and CPU time for solving CILP with a time limit of 5000 s using CPLEX. The gap parameter 'Gap(%)' for SUB + MST and SUB + MI were calculated by $\frac{Obj-LB}{Obj}\%$. The lower bound obtained by SUB + MI is consistently superior to that of SUB + MST. However, its computation time is significantly larger. To solve the matroid intersection problems, we used the algorithm of [9]. We believe an efficient implementation of the matroid intersection algorithm and good quality fast heuristics for solving Max-ECP could result in better running times. Nevertheless, we believe that both these algorithms are useful in developing specialized branch and bound algorithms taking advantage of the strengths of each in a hybrid way.

As the table shows, the heuristic algorithms LS, TT, and TS produced good quality solutions. The performance of TT and TS are somewhat similar, but TS seems slightly better. Interestingly, the solutions obtained by LS for the problems (100, 300, 448), (100, 500, 1247) and (100, 500, 2495) are better than that of TT and TS. This may appear counterintuitive, but note that LS uses a special starting solution generated by SUB + MST. This suggests that the solution produced by SUB + MST is a good candidate starting solution and could be used to initiate any local search, including TT and TS or should be considered as one of the starting solutions in a multi-start version of TT and TS. Detailed analysis of various fine-tuning mechanisms and enhancements of TS and TT are beyond the scope of this preliminary experimental study. Such a study will be interesting, especially when good practical applications warrant that. The optimality of the problems marked ('#') were not achieved within the time limit of 5000 s provided to CPLEX and we recorded the best upper bound. It may be noted that the tabu search heuristic was able to find a feasible solution in all cases. For the problem (100, 500, 3741) tabu search found a heuristic solution in 35 s but CPLEX could not find a feasible solution in 5000 s.

Table 2
Type 1 problems—Feasible.

n	m	p	LB-MST			LB-MI			Heuristics						Opt.	
			LB	CPU	Gap(%)	LB	CPU	Gap(%)	LS		TT		TS		Obj	CPU
									UB	CPU	UB	CPU	UB	CPU		
50	200	199	701.089	0.156	0.98	702.793	10.453	0.74	708	0.157	735	2.172	711	2.563	708	41.22
50	200	398	739.838	0.204	3.92	757.816	10.828	1.58	797	0.265	789	2.594	785	2.484	770	51.08
50	200	597	782.67	0.453	14.65	807.745	51.203	11.91	–	0.438	1 044	2.609	1086	2.141	917	42.93
50	200	995	835.68	0.500	36.88	877.495	51.641	33.72	1424	0.625	1 721	1.984	1629	2.531	1324	162.59
100	300	448	3893.48	1.844	3.65	3991.18	301.601	1.23	4102	1.906	4 316	14.156	4207	13.859	4041	801.51
#100	300	897	4508.16	1.796	–	4624.24	418.613	–	–	3.844	–	11.907	–	13.125	6523	5000
100	500	1247	4124.53	3.297	3.52	4165.68	794.078	2.56	4293	4.703	4 913	28.985	4539	38.782	4275	1364.66
#100	500	2495	4701.87	3.125	–	4805.40	753.453	–	6603	6.375	7 959	31.031	6812	37.422	6653	5000
#100	500	3741	4743.83	3.735	–	4871.27	781.235	–	–	8.641	10 066	28.407	8787	32.719	–	5000

Table 3
Type 1 problems—Feasibility unknown.

n	m	p	LB-MST		LB-MI		Heuristics					
			LB	CPU	LB	CPU	LS		TT		TS	
							vio	CPU	vio	CPU	vio	CPU
100	300	1 344	4 520.42	2.313	4 681.27	349.72	23	3.187	14	16.922	13	14.829
100	500	6 237	4 724.35	4.828	4 968.99	733.25	23	10.203	14	32.156	11	33.219
100	500	12 474	4 624.59	6.984	5 194.67	921.39	49	15.141	41	22.204	41	32.063
200	600	1 797	1 111.6	13.484	11 425.8	4 448.64	15	32.156	2	186.408	2	158.721
200	600	3 594	11 896.9	11.563	12 487	5 456.88	74	78.204	65	193.845	67	153.83
200	600	5 391	11 953.7	12.453	12 873.2	5 947.77	177	68.501	149	178.126	149	175.471
200	800	3 196	17 428.8	17.313	17 922.6	6 640.04	5	52.313	1	246.845	2	230.409
200	800	6 392	19 061.3	19.078	19 705.7	8 193.83	63	88.36	32	298.955	39	214.643
200	800	9 588	19 229.6	19.797	20 684.8	8 429	126	139.532	105	253.689	95	212.659
200	800	15 980	18 778.1	25.609	20 226.9	9 020.98	189	173.626	180	239.283	178	229.8
300	800	3 196	28 617.2	36.126	30 190.1	19 021.5	89	274.064	61	569.16	63	524.788
300	1000	4 995	39 407.2	47.735	40 732.7	26 828.3	61	311.215	39	940.771	38	663.649
300	1000	9 990	40 748.4	46.422	42 902.5	28 421.5	231	670.119	191	919.522	207	756.104
300	1000	14 985	40 729.9	57.297	44 639.1	27 509.5	355	755.3	342	1188.62	351	835.011

Computational results on the remaining Type 1 problems are summarized in Table 3. For these problems, CPLEX and our heuristics failed to produce a feasible solution. We give the number of violated conflict pairs in the column 'vio' for each heuristic, along with the lower bound values. We are glad to provide our test problems for anyone interested in performing further computational study on MSTC.

Table 4 summarizes the results on type 2 problems, where the problems are known to be feasible. LS, TT and TS returned optimal solutions for every problem in this category, except one. Optimality was verified using CPLEX. Unlike Table 2, where the problems were random, CPLEX had better running time on many problems in this class, compared to the heuristics but for some problems it took more time. In fact CPLEX detected feasibility and optimality in the preprocessing stage itself. This may be because of the way we forced feasibility for this class of problems which generated large number of bridges and tabu search/tabu thresholding could not take advantage of this while CPLEX could. The SUB + MI lower bound is significantly superior to SUM + MST lower bound on this class problems as well. However, as observed and discussed earlier, the computation time for our implementation of SUB+ MI is not very attractive. For the problem (200, 800, 62 625) CPLEX failed to terminate with an optimality certificate within our time limit. For this problem, the objective function value given is the best solution obtained and the CPU time is the time taken to reach this solution.

7. Concluding remarks

In this paper, we considered the problem MSTC and established various complexity results. In particular, we showed when the graph G is a cactus but the conflict graph is arbitrary, the feasibility version of MSTC is polynomially solvable whereas the optimization version is NP-hard. This is somewhat surprising since the problem is strongly NP-hard on a general graph with conflict graph being a collection of 2-paths, a very simple class of graphs. Interestingly, by adding another edge to each of these 2-paths, the problem becomes polynomially solvable. In general, we showed that MSTC is polynomially solvable when the conflict graph is a collection of disjoint cliques or some extensions of such graphs. This property is exploited to generate strong lower bounding schemes and efficient feasibility tests. Another significant outcome of this study is the development of an approximation algorithm for Max-ECP, improving the best known performance ratio for the problem. Further, our result shows that any improvement in the approximation ratio for the maximum clique problem will improve the approximation ratio for Max-ECP. Finally, we present some preliminary experimental results comparing lower bounding schemes, feasibility tests and heuristics based on standard techniques.

Table 4
Type 2 problems.

n	m	p	LB-MST			LB-MI			Opt.		CPU-Heu		
			LB	CPU	Gap(%)	LB	CPU	Gap(%)	obj.	CPU	LS	TT	TS
10	20	86	57.943	0.047	21.70	65.386	0.045	11.64	74	0.187	0.015	0.047	0.093
10	30	182	93.220	0.140	51.45	132.915	0.641	30.77	192	0.813	0.093	0.078	0.141
10	40	190	102.71	0.109	61.82	144.28	0.766	46.36	269	2.281	0.188	0.076	0.142
10	45	475	67.487	0.031	54.40	88.774	0.797	40.04	148	2.514	0.016	0.047	0.156
50	200	3 903	775.118	1.047	52.62	877.467	81.281	46.37	1 636	3.69	1.328	2.219	2.891
50	200	4 877	698.676	1.719	65.80	887.478	78.015	56.56	2 043	0.51	1.657	2.454	4.234
50	200	5 864	626.918	1.328	73.18	1 030.25	80.781	55.93	2 338	0.74	2.5	2.094	3.422
100	300	8 609	4 043.38	4.407	45.61	5 754.85	636.137	22.59	7 434	10.67	4.766	15.627	17.578
100	300	10 686	3 970.52	3.875	50.17	6 192.29	567.48	22.29	7 968	4.65	5.328	11.891	16.36
100	300	12 761	3 936.27	4.672	51.80	6 758.57	585.84	17.24	8 166	1.38	5.406	16.641	17.266
100	500	24 740	4 289.85	14.687	66.09	5 104.900	1 032.32	59.65	12 652	942.73	30.047	33.236	42.251
100	500	30 886	3 972.68	24.938	64.63	5 078.820	952.713	54.78	11 232	1 333.48	37.454	32.564	36.719
100	500	36 827	3 918.06	34.375	65.87	5 710.770	860.776	50.26	11 481	264.43	49.688	24.798	33.203
200	400	13 660	14 085.8	4.203	20.54	17 245.9	427.185	2.72	17 728	0.38	4.313	62.204	66.298
200	400	17 089	14 067.3	5.625	24.44	18 048.2	443.482	3.06	18 617	0.47	4.828	55.064	84.61
200	400	20 470	13 998.7	9.797	26.86	18 646.2	550.566	2.58	19 140	0.53	10.844	49.422	58.345
200	600	34 504	9 466.060	47.985	54.31	15 393.1	4 880.88	25.69	20 716	657.52	65.313	128.158	180.659
200	600	42 860	9 100.640	67.172	49.51	13 971.5	5 138.65	22.49	18 025	192.67	60.813	114.5	210.584
200	600	50 984	8 734.530	76.032	58.14	16 708.1	5 313.14	19.92	20 864	545.49	73.031	132.704	268.644
*200	800	62 625	16 806.500	115.641	57.87	23 792.300	7 705.11	40.36	39 895	29 370.95	275.674	219.049	256.956
200	800	78 387	15 803.100	140.173	58.05	22 174.200	7 464.64	41.14	37 671	1 008.87	296.83	236.673	233.284
8200	800	93 978	15 470.100	170.313	60.13	24 907.000	7 421.59	35.80	38 798	11 523.08	280.939	197.048	230.878
300	600	31 000	34 154.200	38.266	21.88	42 720.6	2 626.15	2.29	43 721	0.77	40.01	200.532	245.347
300	600	38 216	33 320.100	21.563	24.73	43 486.7	1 710.3	1.76	44 267	0.96	21.266	246.971	337.004
300	600	45 310	32 072.300	11.266	25.54	42 149.0	1 395.09	2.14	43 071	1.16	12.765	155.673	216.799
300	800	59 600	24 384.300	117.876	43.46	36 629.600	19 945.3	15.06	43 125	1.48	165.72	442.581	469.756
300	800	74 500	22 913.200	140.36	45.82	38 069.300	18 248.7	9.98	42 292	1.79	144.016	298.783	419.662
300	800	89 300	21 624.600	41.188	50.98	38 843.000	17 969.2	11.95	44 114	2.13	46.001	405.378	536.819
300	1000	96 590	36 544.700	217.376	48.93	56 048.300	30 428.1	21.68	71 562	32.71	700.583	531.503	659.789
300	1000	120 500	34 380.800	253.658	54.97	58 780.100	27 779.3	23.01	76 345	82.45	729.676	571.191	629.602
300	1000	144 090	33 481.200	317.83	57.55	60 810.800	27 788.1	22.91	78 880	99.38	627.207	616.598	712.103

The experimental results show that MSTC is a computationally challenging problem and we hope this work and the preceding works [4,5] on the topic will generate further interest in theoretical and experimental investigations.

Acknowledgements

We are thankful to the referees and an associated editor for their helpful comments which improved the presentation of the paper.

References

[1] F.K. Hwang, D.S. Richards, P. Winter, The Steiner Tree Problem, North-Holland, New York, 1992.
[2] S.C. Narula, C.A. Ho, Degree-constrained minimum spanning tree, Computer and Operation Reserch 7 (1980) 239–249.
[3] A. Amberg, W. Domschke, S. Voß, Capacitated minimum spanning trees: algorithms using intelligent search, Combinatorial Optimization: Theory and Practice 1 (1996) 9–40.
[4] A. Darmann, U. Pferschy, J. Schauer, Minimal spanning trees with conflict graphs. 2009. Optimization online, http://www.optimization-online.org/DB_HTML/2009/01/2188.html.
[5] A. Darmann, U. Pferschy, S. Schauer, G. Woeginger, J. Paths, Trees and Matchings under Disjunctive Constraints, 2009, *Optimization online*, http://www.optimization-online.org/DB_HTML/2009/10/2422.html.
[6] A.P. Punnen, R. Zhang, Quadratic bottleneck problems, Manuscript, Simon Fraser University, 2009 (submitted for publication).
[7] R. Zhang, A.P. Punnen, Quadratic Bottleneck Optimization Problems, Presentation at CORS, London, Ontario, 2007.
[8] G. Gutin, A.P. Punnen, The Traveling Salesman Problem and Its Variations, Kluwer, Dordrecht, 2002.
[9] C. Brezovec, G. Cornuéjols, F. Glover, Two algorithms for weighted matroid intersection, Mathematical Programming 36 (1986) 39–53.
[10] J. Edmonds, Matroid intersection, Annals of Discrete Mathematics 4 (1979) 39–49.
[11] A. Dessmark, J. Jansson, A. Lingas, E.-M. Lundell, M. Persson, On the approximability of maximum and minimum edge clique partition problems, International Journal of Foundations of Computer Science 18 (2007) 217–226.
[12] S. Khot, Improved inapproximability results for maxclique, chromatic number, and approximate graph coloring, in: Proceedings of the 42nd IEEE symposium on Foundations of Computer Science, FOCS, 2001, pp. 600–609.
[13] T. Feder, Network flow and 2-satisfiability, Algorithmica 11 (1994) 291–319.
[14] B. Aspvall, M.F. Plass, R.E. Tarjan, A linear-time algorithm for testing the truth of certain quantified Boolean formulas, Information Processing Letters 8 (1979) 121–123.
[15] D.J.A. Welsh, Matroid Theory, 1st edition, Academic Press, 1976.
[16] C. Brezovec, G. Cornuéjols, F. Glover, A matroid algorithm and its application to the efficient solution of two optimization problems on graphs, Mathematical Programming 42 (1988) 471–487.
[17] A. Figueroa, A. Goldstein, T. Jiang, M. Kurowski, A. Lingas, M. Persson, Approximate clustering of fingerprint vectors with missing values, in: In Proc. 11th Computing: The Australasian Theory Symposium (CATS), vol. 41, 2005, pp. 57–60.

[18] U. Feige, Approximating maximum clique by removing subgraphs, SIAM Journal on Discrete Mathematics 18 (2005) 219–225.
[19] A. Assad, W. Xu, The quadratic minimum spanning tree problem, Naval Research Logistics 39 (1992) 399–417.
[20] R. Cordone, G. Passeri, 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, 525.
[21] T. Oncan, A.P. Punnen, The quadratic minimum spanning tree problem: a lower bounding procedure and an efficient search algorithm, Computers and Operations Research 37 (2010) 1762–1773.
[22] G. Zhou, M. Gen, Genetic algorithm approach on multi-criteria minimum spanning tree problem, European Journal of Operational Research 114 (1999) 141–152.
[23] N.Z. Shor, Minimization Methods for Non-Differentiable Functions, in: Springer Series in Comoputational Mathematics, Springer, 1985.
[24] F. Barahona, R. Anbil, The volume algorithm: producing primal solutions with a subgradient method, Mathematical Programming 87 (2000) 385–399.
[25] F. Glover, Tabu thresholding: improved search trajectories by non-monotonic search trajectories, ORSA Journal on Computing 7 (1995) 426–442.
[26] J.A. Bennell, K.A. Dowsland, A tabu thresholding implementation for the irregular cutting problem, International Journal of Production Research 37 (1999) 4259–4275.
[27] D. Castelino, N. Stephens, A surrogate constraint tabu thresholding implementation for the frequency assignment problem, Annals of Operations Research 86 (1999) 259–270.
[28] A. Kaveh, A.P. Punnen, Randomized local search and improved solutions for the microarray QAP, Manuscript, SFU Mathematics department, 2008.
[29] F. Glover, Tabu search: part I, ORSA J. Comput. 1 (1989) 190–206.
[30] F. Glover, Tabu search: part II, ORSA J. Comput. 2 (1990) 4–32.
[31] T.L. Magnanti, L.A. Wolsey, Optimal trees, in: M.O. Ball, T.L. Magnanti, C.L. Monma, G.L. Nemhauser. (Eds.), Network Models, in: Handbook in Operations Research and Management Science, vol. 7, North-Holland, Amsterdam, 1995, pp. 503–615.