THE MINIMUM SPANNING TREE PROBLEM WITH CONFLICT CONSTRAINTS AND ITS VARIATIONS

RUONAN ZHANG, SANTOSH N. KABADI, AND ABRAHAM P. PUNNEN

Abstract. We consider the minimum spanning tree problem with conflict constraints (MSTC). It is observed that computing an ϵ-optimal solution to MSTC is NP-hard for any $\epsilon > 0$. For a general conflict graph, 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 is a matroid intersection problem and hence can be solved in polynomial time. We also identify other special cases where MSTC can be solved in polynomial time. These include instances where the conflict graph is a collection of disjoint cliques and a fixed number of stars. Exploiting these polynomially solvable special cases, we derive strong lower bounds and develop an exact branch and bound algorithm. Also various heuristic algorithms are discussed to solve the problem along with preliminary experimental results. As a byproduct of this investigation, we obtained a polynomial time approximation algorithm for the maximum edge clique partitioning problem with performance ratio $O\left(\frac{n(\log\log n)^2}{\log^3 n}\right)$, improving the previously best known bound of $O(n)$.

1. Introduction

The minimum spanning tree problem (MST) is perhaps the most well-studied combinatorial optimization problem. While the MST can be solved in polynomial time by a greedy algorithm, many of its variations such as the Steiner tree problem [18], degree constrained minimum spanning tree problem [21], capacitated minimum spanning tree problem [1] etc. are all NP-hard. Rcently, Darmann, Pferschy and Schauer [9] 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.

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. Thus MSTC is an interesting and relevant problem for further investigation. Recently, Zhang and Punnen [24] used MSTC to develop exact and heuristic algorithms for the *quadratic bottleneck spanning tree problem.*

Key words and phrases. Minimum spanning tree, Matroid intersection, conflict graphs, heuristics, branch and bound, combinatorial optimization.

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

Let $\hat{G} = (\hat{V}, \hat{E})$ be the undirected graph with its vertex set $\hat{V} = E$. Its edge set $\hat{E}$ is defined such that the $(e, f) \in \hat{E}$ if and only if $\{e, f\} \in S$. Then $\hat{G}$ is called the *conflict graph*. Recently Darmann, Pferschy and Schauer [9] 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.

In this paper, we observe that computing a feasible solution or an ϵ-optimal solution to MSTC is NP-hard for any $\epsilon > 0$. When G is a cactus, we show that the feasibility problem is polynomially solvable whereas the optimization version is still NP-hard. 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 [6, 11]. 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)* [10]. 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 [20]. We show that Max-ECP can be solved by a polynomial time approximation algorithm with performance ratio $O\left(\frac{n(\log \log n)^2}{\log^3 n}\right)$, improving the best known performance ratio of $O(n)$ [10] for this problem. Further we observe that MSTC is polynomially solvable when the conflict graph is a collection of disjoint cliques together with a fixed number of stars. We then develop a branch and bound algorithm to solve the problem optimally and 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. A branch and bound algorithm is discussed in Section 5. In section 6 we present the computational results illustrating the efficacy of our exact algorithm, heuristic algorithms, and lower bound schemes. 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

As established by Darmann et al [9], MSTC is NP-hard even if the conflict graph is simply a collection of disjoint 2-edge paths. Further, they showed that the problem is polynomially bounded when the conflict graph is a collection of disjoint edges. The feasibility version of MSTC can be described as follows.

"Given a graph G and a conflict set S, does there exist a conflict free spanning tree of G?"

Interestingly, it can be shown that this feasibility problem itself is NP-hard. To establish this we consider 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.

Theorem 1. [24] *QBSTP is NP-hard. Further, QBSTP can be solved optimally by solving $O(\log n)$ MSTC feasibility problems.*

Corollary 2. *MSTC feasibility problem is NP-complete. Further, for any $\epsilon > 0$, computing an ϵ-optimal solution to MSTC is NP-hard.*

The complexity result of Darmann et al [9] considers a general graph G with 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 where as $\hat{G}$ 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 $(E_1, E_2, \dots, E_k)$ of E where each $E_i = \{e_{1,i}, e_{2,i}, \dots, e_{l_i,i}\}$ 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) $\sum\limits_{e \in T^*} c_e$ is minimum, (or equivalently, $\sum\limits_{e \in X^*} c_e$ is maximum), (optimality).

First of all, we shall show that we can assume, without loss of generality, that $\{e_{p,i}, e_{q,j}\} \in S$ implies that $i \neq j$. For this the following two operations will be helpful.

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 as defined below.

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 $i \in \{1, 2, \dots, k\}$, there exist sets of the type $\{e_{p,i}, e_{q,j}\}$ in S. If there exist more than one such sets then, if their intersection is empty, then 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. If there exists only one such set $\{e_{p,i}, e_{q,j}\}$ in S then exclusion of all the edges $E_i - \{e_{p,i}, e_{q,i}\}$ and deleting $\{e_{p,i}, e_{q,j}\}$ from S 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 [12]. From the results on the 2-SAT problem it will then follow that the feasibility version of the problem can be solved in $O(\max\{|E|, |S|\})$ time [3], while the optimality version of the problem is NP-hard [12]. To show the equivalence, we shall need the following problem.

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

Consider an instance of MSTC on a cactus. We shall assume that $\{e_{p,i}, e_{q,j}\} \in S$ implies that $i \neq j$. The problem can be formulated as G2SAT problem by defining $\bar{n}_{i,j} = e_{i,j}$ and $w_{\bar{n}_{i,j}} = c_{e_{i,j}},\ \forall i, j$, and $\{\bar{n}_{p,i}, \bar{n}_{q,j}\} \in F$ 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: $V = \{0\} \cup \{i_j : i \in \{1, 2, \dots, k\}, j \in \{i, 2, \dots, l_i - 1\}\}$. Associate with each $\bar{N}_i$ a cycle $(0, i_1, i_2, \dots, i_{l_i - 1}, 0)$, 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 $\bar{n}_{p,i}$ of $\bar{N}_i$. Let $c_{e_{p,i}} = w_{\bar{n}_{p,i}}\ \forall p, i$ and let $\{e_{p,i}, e_{q,j}\} \in S$ if and only if $\{\bar{n}_{p,i}, \bar{n}_{q,j}\} \in F$. It is easy to see that $G = (V, E)$ is a cactus and this instance of MSTC is equivalent to G2SAT.

We shall now show that G2SAT is equivalent to the well-studied weighted 2-SAT problem [12], 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 $1(=\text{true})$ or $0(=\text{false})$, 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 $i = 1, \ldots, k$, define $\bar{N}_i = \{\bar{n}_{1,i}, \bar{n}_{2,i}, \bar{n}_{3,i}\}$, where $\bar{n}_{1,i}$ corresponds to x_i, $\bar{n}_{2,i}$ corresponds to $\bar{x}_i$ and $\bar{n}_{3,i}$ is a dummy node. Let $w_{\bar{n}_{1,i}} = w_i$, $w_{\bar{n}_{2,i}} = 0$ and $w_{\bar{n}_{3,i}} = -M$ (where M is a sufficiently large integer).

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 $(\bar{n}_{a,i}, \bar{n}_{b,j})$ 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 $v_q = \bar{x}_j$.

From any solution to the weighted 2-SAT problem, we can construct a solution $\bar{N}^*$ to G2SAT that has the same objective function value and does not contain node $\bar{n}_{3,i}$ for any i as follows: For each $i = 1, \ldots, k$, if $x_i = 1$ then set $\bar{n}_{1,i} \in \bar{N}^*$; if $x_i = 0$ then set $\bar{n}_{2,i} \in \bar{N}^*$. Similarly, we can associate with any solution $\bar{N}^*$ to G2SAT does not contain node $\bar{n}_{3,i}$ for any i, a solution to weighted 2-SAT problem with same objective function value. If the given instance of the G2SAT problem is infeasible or an optimal solution to it contains a node $\bar{n}_{3,i}$ for some i, then it follows from the above that the given instance of weighted 2-SAT problem is infeasible.

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 $(\bar{n}_{p,i}, \bar{n}_{q,j}) \in F$, we associate a clause $C_{piqj} = \{x_{pi}, x_{qj}\}$. In addition, we create clauses $\{\{\bar{x}_{pi}, \bar{x}_{qi}\} : i = 1, 2, \ldots, k; \{p, q\} \subseteq \{1, 2, \ldots, l_i\}, p \neq q\}$.

Again equivalence of optimal solution of the two problems is easy to verify and hence the proof is omitted.

Since the weighted 2-SAT problem is NP-hard [12] but its feasibility version can be solved in $O(m)$ [3], we get the following theorem:

Theorem 3. *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, $\{e, f\} \in S$, $\{f, g\} \in S$, implies $\{e, g\} \in S$.

Lemma 4. *The conflict graph* $\hat{G}$ *is a collection of disjoint cliques if and only if the conflict set* S *is transitive.*

Proof. Let $\hat{G}$ be a collection of disjoint cliques and $\{e, f\}, \{f, g\}$ be in S. Thus there exists a path $e - f - g$ in $\hat{G}$. Since $\hat{G}$ is a collection of disjoint cliques, it must contain an edge from e to g. Thus $\{e, g\} \in S$ and hence S is transitive.

Conversely, suppose S is transitive. Let e and f be two nodes in $\hat{G}$ so that there exists a path from e to f. Let $P(e, f) = e - e_1 - e_2 \ldots - e_q - f$ be such a path. By transitivity of S, there must exist an edge (e, e_2) in $\hat{G}$. Applying the transitivity property repeatedly it can be seen that there exists an edge from e to every other node in $P(e, f)$; in particular, an edge from e to f. Thus $\hat{G}$ must be a collection of disjoint cliques. $\square$

We now show that MSTC is polynomially solvable whenever S is transitive. In this case, by Lemma 4, $\hat{G} = (\hat{V}, \hat{E})$ is a collection of disjoint cliques. Let $\hat{G}_1, \hat{G}_2, \ldots, \hat{G}_p$ be the connected components of $\hat{G}$, each one of which 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 [23]. 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 [23]. 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 [11]. The forgoing discussion is summarized in the theorem below:

Theorem 5. *When the conflict graph is a collection of disjoint cliques (equivalently when S is transitive), MSTC can be solved in polynomial time.*

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 [7]. 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 6. *Suppose the conflict graph $\hat{G}$ contains a fixed number of stars which can be identified in polynomial time, such that deletion of the centers of these stars from $\hat{G}$ reduces it to a collection of disjoint cliques, then MSTC on G can be solved in polynomial time.*

Proof. Let k be the number of stars. If the edge in G representing the center of a star in $\hat{G}$ is to be excluded from a tree, we delete the edge from graph G and the corresponding node from $\hat{G}$. If the edge representing the center of a star is to be included in a tree, we change its cost in G to $-M$, where M is a suitably large number and in $\hat{G}$, we delete all the end nodes of the star whose center is this edge. There are 2^k ways one can choose centers of the stars to be included or excluded. In any given such selection, after performing the above operations corresponding to the inclusion/exclusion of these nodes, it can be verified that the resulting conflict graph will be a collection of disjoint cliques. By theorem 5, this problem can be solved in polynomial time. If the optimal solution does not include all the edges of weight $-M$, 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 5. $\square$

Note that in theorem 6, although we fix the number of stars, the number of conflict pairs associated with these stars need not be fixed since each such star can have an arbitrary number of pendant nodes. Figure 1 above gives an example of a conflict graph satisfying conditions of theorem 6.

As a corollary of theorem 6, we have the following.

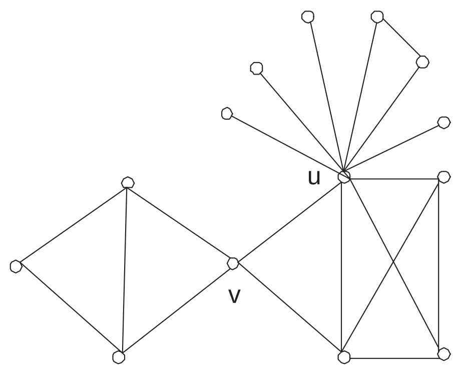

FIGURE 1. Example of a graph satisfying conditions of theorem 6. The nodes u and v are centers of stars.

Corollary 7. *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 6 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) [10, 14, 20]. 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.[10] 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 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 [10] 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 8. *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

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

□

Note that $\delta \leq \varrho$. As in the case of ϱ, the ratio δ could also be $O(n)$. However, the advantage of Lemma 8 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 [10], 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 $E(G^k) \neq \emptyset$ 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 9. *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 $|V(D^k)| \geq 2$ 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

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

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

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

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 $|Q_i^k| \leq \varrho^k \leq \epsilon d^k$ for all $i = 1, 2, \ldots, t$. Thus $|E(Q_i^k)| - |E(Q_i^{k+1})| \leq |Q_i^k \cap D^k|(\epsilon d^k - 1)$ if $Q_i^k \cap D^k \neq \emptyset$ 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

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

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

$$\begin{aligned}
\sum_{i=1}^{t} |E(Q_i^1)| &\leq \sum_{k=1}^{r} d^k(\epsilon d^k - 1) \\
&= \sum_{k=1}^{r} (d^k \epsilon (d^k - 1) + \epsilon - 1) \\
&= 2\epsilon \sum_{k=1}^{r} \frac{d^k(d^k-1)}{2} + r(\epsilon - 1) \\
&= 2\epsilon |E(H)| + r(\epsilon - 1)
\end{aligned}$$

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 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 [10] and hence Theorem 9 is a proper generalization of the corresponding result of [10]. Since the maximum clique problem can be approximated within a factor of $\frac{n(\log\log n)^2}{(\log n)^3}$ [13] we have

Corollary 10. *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 $O(n)$ and corollary 10 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

$$\begin{aligned} \text{ILP:}\quad & \min \sum_{e \in E} c_e x_e \\ & \text{Subject to} \\ & \qquad x \in \mathcal{F} \\ & \qquad x_e + x_f \le 1 \ \forall \ \{e, f\} \in S \qquad (3) \\ & \qquad x_e = 0 \text{ or } 1 \forall e \in E \end{aligned}$$

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

$$\begin{aligned} \text{ILP-Star:}\quad & \min \sum_{e \in E} c_e x_e \\ & \text{Subject to} \\ & \qquad x \in \mathcal{F} \\ & \qquad d_e x_e + \sum_{j \in \hat{V}(e)} x_j \le d_e \ \forall \ e \in \hat{V} \qquad (4) \\ & \qquad x_e = 0 \text{ or } 1 \forall e \in E \end{aligned}$$

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:

$$\begin{aligned} & \min \sum_{e \in E} \sum_{f \in E} d_{ef} x_e x_f \\ & \text{Subject to} \\ & \qquad x \in \mathcal{F} \\ & \qquad x_e = 0 \text{ or } 1 \end{aligned}$$

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 [2, 25] 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 better exact and heuristic algorithms.

Let us now restrict our attention to computing good quality lower bounds for the problem MSTC. We first consider a somewhat straightforward lower bound using Lagrangian relaxation of constraints (3) in ILP. Let λ_{ef} be the Lagrange 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 $A(e)$ be the set of its adjacent nodes. Consider the Lagrangian function

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

From Lagrangian duality, $L(\lambda)$ is a lower bound for the optimal objective function 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 [22] or the volume algorithm [4]. 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:

$$\min \left\{ \sum_{e \in E} \left(c_e + \sum_{f \in A(e)} \lambda_{ef} \right) x_e \right\}$$

Subject to

$$x \in \mathcal{F}.$$

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 5 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

$$\text{ILP-MI:}\quad \min \sum_{e\in E} c_e x_e$$

Subject to

$$x \in \mathcal{F}$$

$$x_e + x_f \le 1 \ \forall\ (e,f) \in E(\tilde{G}) \tag{5}$$

$$x_e + x_f \le 1 \ \forall\ (e,f) \in \Delta \tag{6}$$

$$x_e = 0 \text{ or } 1 \text{ for all } e \tag{7}$$

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

$$\ell(\lambda) = \min\left\{\sum_{e\in E} c_e x_e + \sum_{(e,f)\in\Delta} \lambda_{ef}(x_e + x_f - 1) : x \in \mathcal{F}, x_e + x_f \le 1 \ \forall\ (e,f) \in E(\tilde{G})\right\}$$

$$= -\sum_{(e,f)\in\Delta} \lambda_{ef} + \min\left\{\sum_{e\in E}\left(c_e + \sum_{f\in\bar{A}(e)} \lambda_{ef}\right) x_e : x \in \mathcal{F}, x_e + x_f \le 1 \ \forall\ (e,f) \in E(\tilde{G})\right\}$$

From Lagrangian duality, $\ell(\lambda)$ is a lower bound on the optimal objective function value for each $\lambda \ge 0$. Thus

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

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 [22] or using the volume algorithm [4]. 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:

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

Subject to

$$x \in \mathcal{F}$$

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

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) [10, 14, 20] 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 Figure 2 which consists of a collection of r disjoint 3-edge paths.

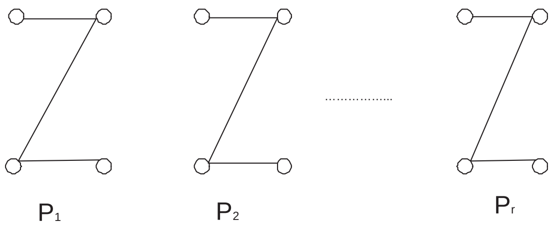

FIGURE 2. Example: Matching heuristic vs Approximate greedy

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 Lagrangian based heuristics. As noted earlier, computing a feasible solution to MSTC itself is NP-hard. Thus, our heuristic algorithms may not compute a feasible solution always. 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 the $e \in E$ which appears in the maximum number of conflict pairs, deleting e from G and all the conflict pairs containing e from S. We repeat this process with the updated G and S, until $S = \emptyset$. If G is still connected, then the total 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 $1 \leq i \leq t_e$. 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 $c(T) = \sum\limits_{e \in T} c_e$, 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 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 in a feasible solution or not, further improvements in terms of reducing the number of conflict violations or improving the objective function value of a feasible solution can be explored by careful and systematic search procedures. Tabu thresholding [17] 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 [5, 8]. Tabu thresholding is a special randomized local search algorithm [19]. 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. The process is continued until a prescribed number of iterations are completed and output the best solution obtained. A formal description of the tabu thresholding algorithm is summarized in algorithm 2. We assume that a procedure $Local - Search(T)$ is available which accepts an input spanning tree T and outputs a locally optimal solution for MSTC with respect to the neighborhood $N(T)$.

4.4. **Tabu search heuristic.** Another way to explore the neighborhood $N(T)$ beyond a local optimum is to employ the tabu search method [15, 16]. 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 is consist 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 find the local optimum 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)

Algorithm 2: Tabu Thresholding Heuristic

```
Input: $T$, $\delta$, $\omega$, Max-Iter;
k=1;
T=Local Search(T);
Best-Solution=T;
while $k \leq$ Max-Iter do
  $i = 1$;
  while $i \leq \delta$ do
    generate a random integer $\alpha$ from $\{1, 2, \dots, \omega\}$;
    find $T^0$: the $\alpha^{\text{th}}$ best solution in $N(T)$;
    if $f(T^0) < f$(Best-Solution) then
      let Best-Solution=$T^0$;
    end if
    let $T = T^0$;
    $i = i + 1$;
  end while
  T=Local Search(T);
  if $f(T) < f$(Best-Solution) then
    let Best-Solution=T;
  end if
  $k = k + 1$;
end while
Output: Best-Solution.
```

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 local optimum $\tilde{T}^0$ in $N_A(T)$ and move to $\tilde{T}^0$. The procedure is repeated until the maximum number of iterations is reached and output the best solution obtained. A formal description of the algorithm is summarized in algorithm 3.

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.

4.6. **MIF Test.** The Lagrangian problem of ILP-MI also allows us to perform some sufficiency tests, which if satisfied, we can declare that the MSTC is infeasible. For example, assume for a choice of Δ, $\tilde{G}$ is a collection of r disjoint cliques $Cl_1,\ Cl_2, \dots Cl_r$. Then in MI(λ), for any x satisfying (7), let $\rho = |\{e :\ x_e = 1\}|$ then $\rho \leq m - \sum\limits_{i=1}^{r} |Cl_i| + r$. Therefore, if $n - 1 > m - \sum\limits_{i=1}^{r} |Cl_i| + r$, we know that x cannot be a spanning tree of G, so MSTC is infeasible. We call this MI-feasibility test (MIF test). The MIF test was very useful in our experimental study in determining infeasibility of several test problems.

Algorithm 3: Tabu Search Heuristic

```
Input: $T$, $L$, Max-Iter;
k=1, Tabu-List=$\emptyset$, Tabu-Size=0;
Best-Solution=T;
while $k \leq$ Max-Iter do
  Find $T^0$: the best solution in $N(T) \cap \mathcal{F}_S$, $T^0 = T + e_i^0 - e_j^0$;
  Find $\tilde{T}^0$: the best solution in $N_A(T) \cap \mathcal{F}_S$, $\tilde{T}^0 = T + \tilde{e}_i^0 - \tilde{e}_j^0$;
  if $f(T^0) = f(\tilde{T}^0)$ or $f(T^0) > f(Best - Solution)$ then
    $T = \tilde{T}^0$;
    Add $(\tilde{e}_i^0, \tilde{e}_j^0)$ to the end of Tabu-List;
    if Tabu-Size<L then
      Tabu-Size=Tabu-Size+1;
    else
      Remove the first pair from Tabu-List;
    end if
  end if;
  if $f(T^0) < f(\tilde{T}^0)$ and $f(T^0) < f(Best - Solution)$ then
    $T = T^0$
  end if
  if $f(T) < f$(Best-Solution) then
    let Best-Solution=T;
  end if
  $k = k + 1$;
end while
Output: Best-Solution.
```

5. Branch and Bound Algorithm

In this section we discuss a very simple branch and bound algorithm for MSTC to compute an exact optimal solution. This also helps us to analyze the quality of our heuristic solutions and gain additional insight into the the average complexity of the problem.

Branching Strategy: Choose a node e of $\hat{G}$ of maximum degree. Note that e is an edge in G. Now the feasible solutions of MSTC can be partitioned in two classes: those spanning trees containing the edge e and those do not contain edge e. A best spanning tree that does not contain the edge e can be obtained by solving MSTC on $G - e$ with conflict graph $\hat{G} - e$. The best spanning tree containing e can be obtained by solving MSTC on the graph $G - \hat{V}(e)$ with conflict graph $\hat{G} - (\hat{V}(e))$. (Note that $\hat{V}(e)$ is a collection of edges in G while it is a collection of nodes adjacent to e in $\hat{G}$.) To force e into the optimal tree on this restricted problem, we make the cost of e to $-M$ where M is a large number. However care must be taken to make necessary adjustments to reflect the original value of the cost of e for calculating lower and upper bounds for this restricted problem.

With this branching strategy, a branch and bound algorithm to solve MSTC is self explanatory. A node in the branch and bound tree is pruned when the lower bound is greater than or equal to a known upper bound. A node can be fathomed if we get an optimal solution to the restricted MSTC at any node. For example, if the conflict graph reduces to a null graph or a collection of disjoint cliques for a restricted MSTC at any node of the search tree, the node can be fathomed.

6. COMPUTATIONAL RESULTS

The lower bound algorithms, heuristics and the branch and bound algorithm are coded in C++ and tested on a Dell PC with 3.40 GHz processor, 2.0 G memory running on Windows XP operation system. We used the public domain compiler Dev C++ for all compilation work. The goals of our preliminary experiments were (1) to identify relative strengths of the two lower bounds; (2) comparative study of the heuristics. The branch and bound algorithm was used to assess the quality of the lower bounds and heuristics. No standard benchmark problems are available for MSTC, so we have generated random test instances– 85 random graphs 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]. The conflict pairs for each graph are generated randomly as well. Two classes of test problems are generated. In the first class, called general problems, the density of conflict pairs ranges from 1% to 15% of the total number of edge pairs. No guarantee of a feasible solution is incorporated in the problem generator. The second class of problems are generated with guaranteed feasibility, with density of conflict pairs ranging from 20% to 30% of the total number of edge pairs.

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

1. Sub+MST: subgradient algorithm solving lagrangian relaxation problem of ILP to compute lower bound;
2. Sub+MI: subgradient algorithm solving lagrangian relaxation problem of ILP-MI to compute lower bound;
3. LS: local search, which uses the solution of Sub+MST as the initial solution;
4. TT: tabu thresholding algorithm;
5. TS: tabu search algorithm;
6. BB: branch and bound algorithm.

For (1) and (2), if the Lagrangian value $l(\lambda)$ $(L(\lambda))$ does not increase for 3 iterations, then we find an edge e such that e is involved in violated pairs for the most number of times and the cost of e is reassigned a very large number. Therefore, it is not likely to be selected as a tree edge in the following iterations.

We set the maximum number of iterations in (1) and (2) to be both 100, and the Lagrangian multipliers are updated using similar strategies. Tabu thresholding and tabu search use the same initial solution, which is generated by solving the following minimum spanning tree problem:

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

For tabu search, we set the maximum number of iterations to be 100 and the length of the tabu list to be 7. For tabu thresholding, $\omega = 15$, Max-Iter $= 10$ and the number of random moves in each iteration is 10, so the total number of iterations is 10×10=100, which is the same as that for tabu search.

Table 1 summarizes experimental results on general problems with small number of conflict pairs, and on special problems with small n and m (denoted by '*'). In the table, 'p' represents the number of conflict pairs, 'LB-MST' is the lower bound obtained by Sub+MST and 'LB-MI' is returned by Sub+MI. 'Heuristics' includes LS, TT and TS, each with the best upper bound found and CPU time. 'Opt.' stands for the optimal objective function value obtained by the branch and bound algorithm. For two problem sizes (10,40) and (50,200), LS did not find a feasible solution.

Assume l_1 is the lower bound obtained by Sub+MI and l_2 by Sub+MST. Then within the same maximum number of iterations, we see that l_1 is superior to l_2 for all the 18 problems in table 1. If the relative difference is calculated as $\sigma = \frac{l_1 - l_2}{l_2}$, then σ grows very fast when the number of conflict pairs are increased based on the same graph. When comparing the heuristics for upper bounds, LS seems weaker in terms of its ability to find feasible solutions compared to TT and TS, but the computational time for LS is relatively low. TS obtained better or same quality solutions as that of TT in 17 out of 18 problems and their CPU time are not quite different. For all the special problems, TT and TS both return the optimal solutions.

Algorithm (1)–(5) have also been tested on larger problems and we did not test the branch and bound algorithm on these larger instances due to significant computational time segments. Table 2 presents results on special problems and Table 3 presents results on general problems. In the tables we use the column 'fea' to indicate the feasibility of the problem. If a feasible solution is obtained by any of the algorithms, 'fea' is "Y"; If the MIF test returns "infeasible", then 'fea' is "N"; Else 'fea' is unknown, which is denoted by '—'. The 'best heuristic' column corresponds to the algorithm which got the smallest upper bound when the problem is feasible, or got the least number of violated constraints when the feasibility is unknown.

Among 49 problems in Table 3, 22 are concluded as "infeasible" by the MIF test in Sub+MI, so the algorithm stopped without costing further computation effort. Therefore, we find that besides providing better lower bound, Sub+MI may be used to determine the feasibility of the problems in a very effective way most of the time. For the 27 feasible problems in table 3, TS is the best for 16 cases and it gradually becomes faster than the others when the problem size grows (even though we haven't showed the details here). For the 18 special problems in table 2, all the heuristics got the same bound while LS uses the least of CPU time.

n	m	p	LB-MST	LB-MI	Heuristics						Opt.
					LS		TT		TS		
					UB	CPU(s)	UB	CPU(s)	UB	CPU(s)	
*10	20	86	57.943	65.386	74	0.015	74	0.046	74	0.093	74
*10	30	182	93.220	132.915	192	0.093	192	0.078	192	0.140	192
*10	40	390	102.710	144.280	—	0.188	269	0.078	269	0.141	269
*10	45	475	67.490	88.774	148	0.016	148	0.047	148	0.156	148
50	200	199	701.089	702.793	708	0.157	735	2.172	711	2.563	708
50	200	398	739.838	757.816	797	0.265	789	2.594	785	2.484	770
50	200	597	782.670	807.745	—	0.438	1044	2.609	1086	2.141	917
*50	200	3903	775.118	877.467	1636	1.328	1636	2.219	1636	2.891	1636
*50	200	4877	698.676	887.478	2043	1.657	2043	2.454	2043	4.234	2043
*50	200	5864	626.918	1030.250	2350	2.500	2338	2.094	2338	3.422	2338
100	300	448	3893.480	3991.180	4102	1.906	4316	14.156	4207	13.856	4041
*100	300	8609	4043.380	5754.850	7434	4.766	7434	15.627	7434	17.578	7434
*100	300	10686	3970.520	6192.290	7968	5.328	7968	11.891	7968	16.360	7968
*100	300	12761	3926.270	6758.570	8166	5.406	8166	16.641	8166	17.266	8166
100	500	1247	4124.530	4165.680	4293	4.703	4913	28.985	4539	38.782	4275
*200	400	13660	14085.800	17245.900	17728	4.313	17728	62.204	17728	66.298	17728
*200	400	17089	14067.300	18048.200	18617	4.828	18617	55.064	18617	84.610	18617
*200	400	20470	13998.700	18646.200	19140	10.844	19140	49.422	19140	58.345	19140

TABLE 1

n	m	p	fea	LB-MI	LB-MST	Best Heuristic		
						Name	UB	Best Vio
100	500	24740	Y	5104.900	4289.850	LS/TT/TS	12652	0
100	500	30886	Y	5078.820	3972.680	LS/TT/TS	12652	0
100	500	36827	Y	5710.770	3918.060	LS/TT/TS	11232	0
200	600	34504	Y	15393.100	9466.060	LS/TT/TS	20716	0
200	600	42860	Y	13971.500	9100.640	LS/TT/TS	18025	0
200	600	50984	Y	16708.100	8734.530	LS/TT/TS	20864	0
200	800	62625	Y	23792.300	16806.500	LS/TT/TS	39895	0
200	800	78387	Y	22174.200	15803.100	LS/TT/TS	37671	0
200	800	93978	Y	24907.000	15470.100	LS/TT/TS	38798	0
300	600	31000	Y	42720.600	34154.200	LS/TT/TS	43721	0
300	600	38216	Y	43486.700	33320.100	LS/TT/TS	44267	0
300	600	45310	Y	42149.000	32072.300	LS/TT/TS	43071	0
300	800	59600	Y	36629.600	24384.300	LS/TT/TS	43125	0
300	800	74500	Y	38069.300	22913.200	LS/TT/TS	42292	0
300	800	89300	Y	38843.000	21624.600	LS/TT/TS	44114	0
300	1000	96590	Y	56048.300	36544.700	LS/TT/TS	71562	0
300	1000	120500	Y	58780.100	34380.800	LS/TT/TS	76345	0
300	1000	144090	Y	60810.800	33481.200	LS/TT/TS	78880	0

TABLE 2. Special Problems

n	m	p	fea	LB-MI	LB-MST	Best Heuristic		
						Name	UB	Best Vio
50	200	995	Y	877.495	835.680	LS	1424	0
50	200	1990	—	913.826	808.160	TT/TS	—	14
50	200	2985	—	992.214	768.781	TS	—	32
100	300	897	—	4624.240	4508.160	TS	—	2
100	300	1344	—	4681.270	4520.420	TS	—	13
100	300	2242	—	5030.440	4528.140	TT	—	45
100	300	4484	N					
100	300	6726	N					
100	500	2495	Y	4805.400	4701.870	LS	6603	0
100	500	3741	Y	4871.270	4743.830	TS	8787	0
100	500	6237	—	4968.990	4724.350	TS	—	12
100	500	12474	—	5194.670	4624.590	TS	—	40
100	500	18711	—	5105.910	4450.330	TT	—	129
200	400	798	—	17664.300	16079.300	TT	—	17
200	400	1596	N					
200	400	2394	N					
200	400	3990	N					
200	400	7980	N					
200	400	11970	N					
200	600	1797	—	11425.800	11011.600	TT/TS	—	2
200	600	3594	—	12487.000	11896.900	TT	—	65
200	600	5391	—	12873.200	11953.700	TS	—	149
200	600	8985	—	13501.300	11770.500	TS	—	280
200	600	17970	N					
200	600	26955	N					
200	800	3196	—	17922.600	17428.800	TT	—	1
200	800	6392	—	19705.700	19061.300	TT	—	32
200	800	9588	—	20684.800	19229.600	TS	—	95
200	800	15980	—	20226.900	18778.100	TS	—	178
200	800	31960	—	21458.500	18705.100	TT/TS	—	517
200	800	47940	N					
300	600	1797	N					
300	600	3594	N					
300	600	5391	N					
300	600	8985	N					
300	600	17970	N					
300	600	26955	N					
300	800	3196	—	30190.100	28617.200	TT	—	61
300	800	6392	—	32012.600	29889.300	TS	—	275
300	800	9588	N					
300	800	15980	N					
300	800	31960	N					
300	800	47940	N					
300	1000	4995	—	40732.700	39407.200	TS	—	38
300	1000	9990	—	42902.500	40748.400	TT	—	191
300	1000	14985	—	44639.100	40729.900	TT	—	342
300	1000	24975	—	48233.300	41093.200	TS	—	618
300	1000	49950	N					
300	1000	74925	N					

TABLE 3. General Problems

7. Concluding Remarks

In this paper, we considered the problem MSTC and established various complexity results. Polynomially solvable special cases are identified. Various heuristic algorithms to solve the general problem are proposed along with an exact algorithm and preliminary implementation results. It would be interesting to establish approximability results for specially structured problems.

Many of the results discussed in the paper extends in a natural way to matroids.

References

[1] Amberg A, Domschke W and Voß S, Capacitated minimum spanning trees: algorithms using intelligent search. *Combinatorial Optimization: Theory and Practice, Vol. 1(1996), pp. 9-40.*

[2] Assad A and Xu W, The quadratic minimum spanning tree problem. *Naval Research Logistics, Vol. 39(1992), pp. 399-417.*

[3] Aspvall B, Plass M R, Tarjan R E, A linear-time algorithm for testing the truth of certain quantified Boolean formulas. *Information Processing letters, Vol. 8(1979), pp. 121-123.*

[4] Barahona F and Anbil R, The volume algorithm: producing primal solutions with a subgradient method. *Mathematical Programming, Vol. 87(2000), pp. 385-399.*

[5] Bennell J A and Dowsland K A, A tabu thresholding implementation for the irregular cutting problem. *International Journal of Production Research, Vol 37(1999), pp. 4259-4275.*

[6] Brezovec C, Cornu*éjols* G and Glover F, Two algorithms for weighted matroid intersection. *Mathematical Programming, Vol. 36(1986), pp. 39-53.*

[7] Brezovec C, Cornu*éjols* G and Glover F, A matroid algorithm and its application to the efficient solution of two optimization problems on graphs. *Mathematical Programming, Vol. 42(1988), pp. 471-487.*

[8] Castelino D and Stephens N, A surrogate constraint tabu thresholding implementation for the frequency assignment problem. *Annals of Operations Research, Vol. 86(1999), pp. 259-270.*

[9] Darmann A, Pferschy U and Schauer J, Minimal spanning trees with conflict graphs. Optimization online, 2009.

[10] Dessmark A, Jansson J, Lingas A, Lundell E-M, and Persson M, On the approximability of maximum and minimum edge clique partition problems. *International Journal of Foundations of Theoretical Computer Science, Vol. 18 (2007) pp. 217-226.*

[11] Edmonds J, Matroid intersection. *Annals of Discrete Mathematics, Vol. 4(1984), pp. 338-346.*

[12] Feder T, Network flow and 2-satisfiability. Algorithmica Vol. 11, (1994) pp.291-319.

[13] Feige U, Approximating maximum clique by removing subgraphs. *SIAM J. Discrete Math. Vol. 18 (2004), pp. 219-225.*

[14] Figueroa A, Goldstein A, Jiang T, Kurowski M, Lingas A, and Persson M, Approximate clustering of fingerprint vectors with missing values. *Proc. 11th Computing: The Australian Theory Symposium (CATS 2005), pp. 57-60.*

[15] Glover F, Tabu search: part I. *ORSA Journal on Computing, Vol. 1 (1989), pp. 190-206.*

[16] Glover F, Tabu search: part II. *ORSA Journal on Computing, Vol. 2 (1990), pp. 4-32.*

[17] Glover F, Tabu thresholding: improved search trajectories by non-monotonic search trajectories. *ORSA Journal on Computing, Vol. 7 (1995), pp. 426-442.*

[18] Hwang F K, Richards D S and Winter P, The steiner tree problem. North-Holland, New York (1992).

[19] Kaveh A and Punnen A P, Randomized local search and improved solutions for the microarray QAP, Manuscript, SFU Mathematics department, 2008.

[20] Khot S, Improved inapproximability results for MaxClique, chromatic number, and approximate graph coloring. *Proc. 42th Annual Symposium on Foundations of Computer Science (FOCS 2001), pp. 600-609.*

[21] Narula S C and Ho C A, Degree-constrained minimum spanning tree. *Comput. Oper. Res. Vol. 7 (1980), pp. 239-249.*

[22] Shor N Z, Minimization methods for non-differentiable functions. *Springer Series in Comoputational Mathematics, Springer, (1985).*

[23] Welsh D J A, Matroid Theory. *Academic Press, 1st edition (1976)*

[24] Zhang R and Punnen A P, Combinatorial optimization problems with quadratic bottleneck objective function, Manuscript, Simon Fraser University, 2009.

[25] Zhou G and Gen M, Genetic algorithm approach on multi-criteria minimum spanning tree problem. *European Journal of Operational Research, Vol. 114(1999), pp. 141-152.*

Department of Mathematics, Simon Fraser University Surrey, Central City, 250-13450 102nd AV, Surrey, British Columbia,V3T 0A3, Canada
E-mail address: rza1@sfu.ca

Faculty of Business Administration, University of New Brunswick, Fredericton, New Brunswick, Canada
E-mail address: kabadi@unb.ca

Department of Mathematics, Simon Fraser University Surrey, Central City, 250-13450 102nd AV, Surrey, British Columbia,V3T 0A3, Canada
E-mail address: apunnen@sfu.ca