On the Crossing Spanning Tree Problem

Vittorio Bilò[1], Vineet Goyal[2 ⋆], R Ravi[2 ⋆], and Mohit Singh[2 ⋆]

[1] Dipartimento di Informatica Università di L'Aquila. Via Vetoio, Coppito 67100 L'Aquila. Italy. bilo@di.univaq.it

[2] Tepper School of Business, Carnegie Mellon University, Pittsburgh PA 15213 {vgoyal,ravi,mohits}@andrew.cmu.edu

Abstract. Given an undirected n-node graph and a set $\mathcal{C}$ of m cuts, the *minimum crossing tree* is a spanning tree which minimizes the maximum crossing of any cut in $\mathcal{C}$, where the crossing of a cut is the number of edges in the intersection of this cut and the tree. This problem finds applications in fields as diverse as Computational Biology and IP Routing Table Minimization.

We show that a greedy algorithm gives an $O(r \log n)$ approximation for the problem where any edge occurs in at most r cuts. We also design a randomized algorithm that gives a tree T with crossing $O(\log n \cdot \text{OPT} + \log m)$ w.h.p., where OPT is the minimum crossing of any tree. We then demonstrate that the problem remains NP-hard even when G is complete.

Our greedy analysis extends the traditional one used for set cover. The randomized algorithm rounds a LP relaxation of the natural IP formulation of the problem, in stages.

1 Introduction

The Minimum Crossing Spanning Tree (MCST) problem is a generalization of the minimum degree spanning tree problem, which has been widely studied [10]. In an instance of the MCST problem, we are given an undirected graph G and a family of cuts $\mathcal{C}$=$\{C_1, \ldots, C_m\}$. The problem is to find a spanning tree T which minimizes the maximum crossing of any cut, where the crossing of a cut is the number of edges of T going across that cut. If the family of cuts $\mathcal{C}$ is equal to $\{(v, V \setminus \{v\}) : v \in V\}$, then the crossing of a cut in any spanning tree T is exactly equal to the degree of the corresponding vertex in T. Thus, the MCST problem reduces to finding a spanning tree such that the maximum degree of a vertex is minimized. Since, the minimum degree spanning tree problem is NP-hard [9], MCST is also NP-hard.

1.1 Motivation : Chimerism in Physical Mapping

This problem finds important application in the *physical mapping* problem in computational biology. The *physical mapping* problem of the human genome project is to reconstruct the relative position of fragments of DNA along the genome from information on their pairwise overlap.

⋆ Supported in part by NSF grant CCR-0105548 and ITR grant CCR-0122581 (The ALADDIN project).

In an instance of the physical mapping problem, one has a collection of clones and a set of short genomic inserts (called *probes*). A probe defines a single location where a given subset of clones coincide. For each probe/clone pair, it can be determined whether the clone contains the probe as a subsequence using biological techniques. The problem is to construct the order in which the probes would occur along the original chromosome that is consistent with the given the probe/clone incidence matrix. This can be done efficiently if there is no *chimerism*. *Chimerism* is the result of concatenating two or more clone from different parts of the genome, producing a chimeric clone -one that is no longer a simple substring of the chromosome. More formally, the problem is as follows: Given a probe-clone incidence matrix A, with rows indexed by probes and columns by clones, and the entry a_{ij} is 1 iff probe i occurs in clone j. If there is no chimerism, then the problem is reduced to finding a permutation of rows so that ones in each column are consecutive (called as 1-C1P) and this can be solved efficiently in polynomial time [2]. However, in the presence of chimerism, the problem is more difficult. Then, we need to find a permutation π of rows, such that each column has at most k blocks of consecutive ones (called as k-consecutive ones property or k-C1P), if the chimeric clones are a concatenation of at most k clones. The decision version of this problem i.e *"Does a given 0-1 matrix have the k-consecutive ones property?"* has been proved to be NP-complete in [7].

1.2 k-C1P and Vector TSPs

A classical way to solve the k-C1P problem is to reduce it to a particular multidimensional TSP problem called the Vector TSP (vTSP). This problem is defined on a complete graph $G = (V, E)$, where each edge $e \in E$ is assigned an m-dimensional cost $c : E \rightarrow \{0, 1\}^m$. The cost of a tour T in G is the m-dimensional vector $c(T) = \sum_{e \in E(T)} c(e)$ and the objective is to minimize $\|c(T)\|_\infty$.

The reduction from k-C1P to vTSP is straightforward. Each row of A becomes a node in G and the cost assigned to edge $e = (i, j)$ is set to the XOR-vector between the two rows a_i and a_j. Now, let π be the permutation induced by a solution T of vTSP, and let $b(A^\pi)$ be the maximum number of blocks of consecutive ones in A^π. Then, we have that $b(A^\pi) = \frac{\|c(T)\|_\infty}{2}$. Solving the vTSP problem is NP-hard by this reduction from the 2-C1P problem. However, since the hamming distance obeys the triangle inequality, it is possible to use the standard Euler Tour shortcutting technique in order to compute a $2r$-approximate solution once given an r-approximation to the related Vector MST problem (vMST).

The vMST can be formulated as the minimum crossing spanning tree problem on a complete graph G. A column j of A can be seen as a cut $C_j = (V_j, V \setminus V_j)$ defined on G by setting $V_j = \{v_i \in V | a_{ij} = 0\}$. The cost of edge $e = (i, j)$ is as before the XOR-vector between a_i and a_j i.e. $c(e)$ is a 0-1 vector, where the l^{th} entry corresponding to a cut C_l is 1 iff the edge (i, j) crosses C_l. Here, the terminology that an edge e crosses a cut C is used interchangeably with $e \in C$. For any tree T, let $c(T) = \sum_{e \in E(T)} c(e)$. The i^{th} entry of the vector $c(T)$ is exactly the number of edges of T crossing the cut C_i. Thus, the minimum crossing spanning tree minimizes $\|c(T)\|_\infty$.

1.3 Motivation : IP routing

Another useful application of the MCST problem can be found in [4] where it is shown that the an efficient solution for the min-k-C1P can be used to obtain an good approximation for the Interval Routing problem: given a set of IP routing tables sharing the same host space, the problem is to reassign the IP addresses to the hosts in order to minimize the maximum size of any IP routing table.

This IP routing table minimization problem, MIN-IP for short, can be formalized as follows. We are given a set $R = \{r_1, \ldots, r_n\}$ of n routers and a set $H = \{h_1, \ldots, h_m\}$ of m destination hosts. Each router $r_j \in R$ has a degree δ_j, that is δ_j outedges, and a routing table specifying which of the outedges to take for every host. The problem is to choose the IP addresses of the m hosts and construct the n IP routing tables so as to minimize the maximum size of a table, that is the maximum number of used entries in a table.

In [4] it is shown that, given any ρ-approximation algorithm for the problem of determining a row permutation that minimizes the maximum number of blocks (of ones) in a boolean matrix A, an efficient $2\rho \cdot \log m$-approximation algorithm exists for MIN-IP, which exploits a matrix representation of the instances of the problem. The former problem is an instance of k-C1P and our previous discussion illustrates how our methods can be used.

Similar applications can be found also in designing interval routing schemes as proposed in [5, 6].

1.4 Problem Definition

The MCST problem is formally defined as follows : Given a graph $G = (V, E)$ with n nodes and a family of cuts $\mathcal{C} = \{C_1, \ldots, C_m\}$, *the minimum crossing tree* is a spanning tree T, which minimizes the maximum crossing of any cut. The crossing of a cut C_i is defined as $|E(T) \cap C_i|$.

MCST is NP-hard because it is a generalization of the minimum degree spanning tree problem [10], as noted before. Infact, we prove that MCST is NP-hard even if the input graph is complete. Note that, the minimum degree spanning tree problem (where the objective is to find a spanning tree such that the maximum degree of a vertex is minimized), is trivial if the given graph is complete.

1.5 Previous Work and Our Results

As observed earlier, the minimum degree spanning tree problem is a special case of the MCST problem. A local search technique [10] gives a spanning tree with maximum degree $O(\Delta^* + \log n)$ in $O(n^3)$ time where Δ^* is the maximum degree of the optimal tree. They also improve the above algorithm to get a spanning tree with maximum degree at most $\Delta^* + 1$ which is the best guarantee unless $P = NP$.

The $vMST$ problem has been considered by Greenberg and Istrail [8]. They give solution of cost $O(r \cdot OPT + \log n)$, where each edge in the undirected graph occurs in at most r cuts. They adapt the local search techinque in [10] but do not show that their algorithm runs in polynomial time. They bound the running time of their algorithm by

$O(m^{\log r})$ local search iterations which for $r = \omega(1)$ is not polynomial. In a preliminary version of our paper [3], it is shown that a randomized algorithm gives a spanning tree which has a maximum crossing $O((\log n + \log m)(OPT + \log n))$ with high probability for the case of complete graphs. This algorithm is based on formulating several multi-commodity flow problems as linear programs and rounding them. It is not hard to generalize the technique in [3], that solves a different LP in each iteration, for the case of arbitrary graphs. However, the method described in this paper is simpler and solves only a single LP unlike [3]. We formulate a single linear programming relaxation and use the same LP solution to round in phases. Furthermore, this technique works for arbitrary graphs and not just complete graphs. We prove the following guarantee for this algorithm.

Theorem 1.1. *There is a randomized LP rounding based algorithm, which given a graph G and a family of cuts $\mathcal{C}=\{C_1, \ldots, C_m\}$ gives a spanning tree T such that crossing for any cut $C_i \in \mathcal{C}$ is $O(\log n \cdot \mathit{OPT} + \log m)$ with high probability, where OPT is the maximum crossing of an optimal tree.*

We also give a greedy algorithm for the problem and using a combinatorial argument, prove the following guarantees.

Theorem 1.2. *Given a graph $G = (V, E)$ and a family of m cuts $\mathcal{C}=\{C_1, \ldots, C_m\}$, a greedy algorithm for MCST problem gives a spanning tree T which crosses any cut in $\mathcal{C}$ $O(r \cdot \log n)$ times the maximum crossing of an optimal tree, where an edge e occurs in at most r cuts in $\mathcal{C}$.*

Although the minimum degree spanning tree problem is trivial on complete graphs, surprisingly, the MCST problem remains difficult even for this special case. We show that the decision version of even this version of the MCST problem is NP-complete.

Theorem 1.3. *Given a complete graph G, set of cuts $\mathcal{C}$ and a positive integer k, the problem of determining whether there exists a spanning tree of G which crosses any cut in $\mathcal{C}$ at most k times is NP-complete.*

The rest of the paper is organized as follows. In Section 2, we describe a greedy algorithm for the MCST problem and prove Theorem 1.2. In Section 3, we give a randomized algorithm and prove the guarantees of Theorem 1.1. In Section 4, we show that the MCST problem is NP-hard even for complete graphs.

2 A Greedy Algorithm

In this section, we show that the greedy algorithm gives an $O(r \cdot \log n)$ approximation for the MCST problem where r is defined as $\max_{e \in G} |\{C \in \mathcal{C}\colon e \in C\}|$. Given any subgraph H, the maximum number of times H crosses any cut in $\mathcal{C}$ is denoted by $Cross(H, \mathcal{C})$.

Greedy Algorithm:

$F \leftarrow \phi$
while F is not a tree
do
 Let e' be an edge which minimizes $Cross(F \cup e, \mathcal{C})$
 over all edges $e \in G$ joining two components of F.

 $F \leftarrow F \cup e'$
od

Let the solution returned by the greedy algorithm be T_g and let $l = Cross(T_g, \mathcal{C})$. We can divide the running of the greedy algorithm in l phases. The i^{th} phase of the algorithm is the period when $Cross(F, \mathcal{C}) = i$. Let k_i denote the number of components in F when the i^{th} phase ends. Let M_i be the cuts which are crossed by i edges at the end of i^{th} phase and $m_i = |M_i|$. We now give a lower bound for the MCST problem.

Lemma 2.1. *Given any $S \subset \mathcal{C}$, let k be the number of components formed after removing the edges from G of all cuts in S. Then*

$$opt \geq \frac{k-1}{|S|}$$

Proof. Any spanning tree of G must choose at least $k-1$ edges to join the k components formed after removing the edges of cuts in S. Each of these $k-1$ edges crosses at least one of the cuts in S. Hence, the average crossing of such a cut in S is at least $\frac{k-1}{|S|}$.

Now, we prove Theorem 1.2.

The proof of Theorem 1.2 Consider the running of the algorithm in the i^{th} phase. Crossing number of at least m_i cuts increases by 1 in the i^{th} phase. Each edge can increase the crossing number of at most r cuts. Hence, in the i^{th} phase we must include at least $\lceil \frac{m_i}{r} \rceil$ edges in F. Every edge, when included in F, reduces the number of components in F by exactly one. Therefore, we have the following inequality

$$k_i \leq k_{i-1} - \frac{m_i}{r} \tag{1}$$

When the i^{th} phase ends, every edge joining two components of F must cross at least one of the cuts in M_i, else the greedy algorithm would choose such an edge in the i^{th} phase. Applying Lemma 2.1, we get the for each i,

$$opt \geq \frac{k_i - 1}{m_i} \tag{2}$$

Using (1) and (2), we have that for each i,

$$k_{i-1} - k_i \geq \frac{k_i - 1}{r \cdot opt} \tag{3}$$

Using $k_i \geq 2$ for each $i \leq l-1$ and $k_{l-1} > k_l$, we have for each i,

$$
\begin{aligned}
k_{i-1} - k_i &\geq \tfrac{k_i}{2r \cdot opt} \\
\Rightarrow k_{i-1} &\geq k_i\left(1 + \tfrac{1}{2r \cdot opt}\right) \\
\Rightarrow k_0 &\geq k_l\left(1 + \tfrac{1}{2r \cdot opt}\right)^l
\end{aligned}
$$

As, $k_0 = n$ and $k_l = 1$, we get that

$$n \geq \left(1 + \frac{1}{2r \cdot opt}\right)^l$$

$$\Rightarrow \log n \geq l \log\left(1 + \frac{1}{2r \cdot opt}\right)$$

Using, $\log(1 + x) \geq x - \frac{x^2}{2}$ for $0 < x < 1$ and $r \cdot opt \geq 1$ we get

$$\log n \geq l\left(\frac{1}{2r \cdot opt}\left(1 - \frac{1}{4r \cdot opt}\right)\right) \geq l \frac{1}{4r \cdot opt}$$

$$\Rightarrow \quad l \leq 4r \log n \cdot opt$$

Hence, the greedy algorithm is a $O(r \log n)$ approximation. □

3 A Randomized Algorithm

In this section, we prove that a randomized algorithm gives a spanning tree, whose crossing is at most $O(\log n \cdot \text{OPT} + \log m)$ with high probability. The technique used here is motivated by [1]. We first solve a linear programming relaxation for the problem and round the variables in phases. Let $\hat{z}$ be the value of the optimum LP solution. We show that after $O(\log n)$ phases, we obtain a spanning tree in which no cut is crossed more than the $O(\hat{z} \cdot \log n + \log m)$ with very high probability.

Linear Program We formulate the following integer program for the crossing spanning tree problem where we impose the condition that the set of edges chosen is a spanning tree using the constraint that every cut must be crossed.

$$
\begin{array}{llll}
\min & z & & \\
\text{s.t.} & z & \geq \sum_{e \in E \cap C} x_e & \forall C \in \mathcal{C} \\
& \sum_{e \in \delta(S)} x_e & \geq 1 & \forall S \subset V \\
& & x_e \in \{0, 1\} & \forall e \in E
\end{array}
$$

Relax the integrality constraints to get a linear programming relaxation of the problem. Let $(\hat{z}, \hat{x})$ denote the optimal solution to the linear programming relaxation.

We round this fractional solution in stages. In stage i, choose a set of edges F_i in the following manner. For each edge $e \in E$, include $e \in F_i$ randomly and independently with probability $\hat{x}_e$. Lemma 3.1 [1] states that after $O(\log n)$ stages, the edges selected form a connected graph, with high probability.

We include the proofs for completeness. Let us denote the set of components in $G_i = \bigcup_{j=1}^{i} F_j$ by C_i. We call a stage successful if either $|C_i| = 1$ or $|C_i| \leq \frac{9}{10}|C_{i-1}|$.

Proposition 3.1 ([1]). *For every i, the conditional probability that stage i is successful, given any set of components in G_i, is at least $\frac{1}{2}$.*

Proof. If G_i is connected, then the claim is trivial. Assume G_i is not connected. Shrink each component of G_i to a singleton vertex and call the constructed multigraph, H. Consider any vertex v in H. An edge e is not included in F_i, with probability $(1 - \hat{x_e})$. Let $\delta(v)$ denote the neighborhood of node v. Hence, the probability that v remains isolated is

$$\prod_{e \in \delta(v)} (1 - \hat{x_e}) \leq exp(- \sum_{e \in \delta(v)} \hat{x_e})$$

Using the fact that $\sum_{e \in \delta(v)} \hat{x_e} \geq 1$, we get that the probability that v is isolated is at most $\frac{1}{e}$. Using linearity of expectation, the expected number of isolated vertices in H is $|H|/e$, and hence with probability at least $\frac{1}{2}$, the number of isolated vertices is at most $2|H|/e$. Hence, the number of connected components in G_i is at most

$$\frac{2|H|}{e} + \frac{1}{2}(|H| - \frac{2|H|}{e}) = (\frac{1}{2} + \frac{1}{e})|H| < 0.9|H|$$

Using that $|H| = C_i$, we get the desired result.

Lemma 3.1 ([1]). *After $t = 40 \log n$ rounds, the probability that G_t is not connected is at most $\frac{1}{n^5}$.*

Proof. Observe that G_t gets connected after at most $\log_{0.9} n < 10 \log n = s$ successes. Let X denote the number of successful rounds in t rounds. Hence, if G_t is not connected, then $X < s$. The probability of this event is no more than the probability that we get at most s heads in t independent tosses of a fair coin. Although, the event that a round i is successful is not independent for different i, we can apply the above bound, since Proposition 3.1 gives a lower bound on success, given any history. We can bound this probability using estimates for tails of binomial distributions. For $t = 40 \log n$, expected number of successful rounds $\mu = 20 \log n$. Thus,

$$Pr(X < s) \leq Pr(|X - \mu| > (\mu - s)) \leq e^{-\frac{s^2}{2\mu}}$$

$$\leq e^{-(10 \log n)^2 / 2 \cdot 10 \log n} \leq \frac{1}{n^5}$$

□

Now we argue in Lemma 3.2 that the maximum crossing of the cuts in the graph obtained after $t = 40 \log n$ stages is small.

Lemma 3.2. *After $t = 40 \log n$, the probability that any of the cuts is crossed more than $(80 \log n \hat{z} + 2 \log m)$ is at most $\frac{1}{n^2}$.*

Proof. Let X_e^i denote the random variable which is 1 if e is selected in the i^{th} round and 0 otherwise. Note that $E(X_e^i) = Pr(X_e^i = 1) = \hat{x}_e$. Also, note that the random variables $\{X_e^i : 1 \leq i \leq t, e \in E\}$ are independent $0 - 1$ variables.

Take any cut $C \in \mathcal{C}$. Let Y_C be the random variable denoting the crossing C after t rounds. Clearly, $Y_C = \sum_{e \in C} \sum_{i=1}^{t} X_e^i$. Hence,

$$E(Y_C) = \sum_{e \in C} \sum_{i=1}^{t} E(X_e^i)$$

$$= \sum_{i=1}^{t} \sum_{e \in E} \hat{x}_e$$

$$\leq \sum_{i=1}^{t} \hat{z} = t\hat{z}$$

As Y_C is a sum of independent Bernoulli trials, applying Chernoff bounds we get

$$Pr(|Y_C - E(Y_C)| > t\hat{z} + 2\log m) < e^{-\frac{(t\hat{z}+2\log m)^2}{2E(Y_C)}}$$

$$< e^{-\frac{(t\hat{z}+2\log m)}{2}}$$

$$< e^{-\log mn^2} = \frac{1}{mn^2}$$

Thus the $Pr(Y_C > 2t\hat{z} + 2\log m) < Pr(Y_C > E(Y_C) + t\hat{z} + 2\log m) < \frac{1}{mn^2}$. Now, applying union bound we have that the probability of any cut being crossed more than $(2t\hat{z} + 2\log m)$ is less than $\frac{1}{n^2}$ as required.

Hence, with high probability, after $t = 40\log n$ rounds, we get a spanning tree in which the crossing of every cut is at most $(80\hat{z} \cdot \log n + 2\log m)$ proving Theorem 1.1. □

Remark 3.1. The technique discussed above can be generalized to give a bicriteria approximation for the following problem: Given a graph $G = (V, E)$, a non-negative cost function c on edges, family of cuts $\mathcal{C} = \{C_1, \ldots, C_m\}$ and a bound B_i for cut C_i for each i, the problem is to find the minimum cost spanning tree which crosses each cut at most B_i times. We formulate the following Linear program

$$\begin{array}{rll} \min & \sum_e c_e x_e & \\ \text{s.t.} & \sum_{e \in E \cap C_i} x_e \leq B_i & \forall 1 \leq i \leq m \\ & \sum_{e \in \delta(S)} x_e \geq 1 & \forall S \subset V \\ & x_e \in \{0,1\} & \forall e \in E \end{array}$$

Rounding the above linear program as before, with probability at least $\frac{1}{2}$, we can get a spanning tree with cost at most $O(\log n)c_{opt}$ and crossing of any cut C_i being $O(B_i \cdot \log n + \log m)$. Here c_{opt} is cost of the optimal spanning tree. We can repeat the algorithm $\log n$ times and choose the best overall outcome to get the same guarantees with probability $1 - o(1)$.

4 MCST for Complete Graphs is NP-Hard

In this section, we consider the MCST problem for complete graphs. We show that the problem is NP-hard even for this special case. In fact, we show that the decision version of the problem is NP-complete.

Clearly, the decision problem is in NP. We reduce the 2-consecutive ones problem, 2-C1P, to MCST. Given a $n \times m$ matrix A, 2-C1P is the problem of determining whether there exists a permutation of rows such that in each column all ones occur in at most 2 consecutive blocks. This problem has been shown to be NP-complete in [8].
Given any arbitrary $n \times m$ matrix A, make a complete graph G over $n+1$ vertices, with one vertex corresponding to each row and a new dummy vertex s. For each column in A, include a cut in $\mathcal{C}$, naturally defined by the column: vertices corresponding to rows whose entry in this column is 1 form one side of the cut. The dummy vertex s is always on the side corresponding to rows with entry 0 in the column corresponding to that cut. Also include in $\mathcal{C}$ singleton cuts, $C_v = (\{v\}, V \setminus \{v\})$ for every vertex in G. For each pair of vertices u and v, include in $\mathcal{C}$ the cut $C_{uv} = (\{u, v\}, V \setminus \{u, v\})$. Finally, let $k = 4$.

We first show that if there exists a permutation of rows, π, such that it has 2-C1 property, then there exists a spanning tree which crosses each cut in $\mathcal{C}$ at most four times. Consider the Hamiltonian path H which starts at s and then traverses the vertices in the order corresponding to permutation π. Each cut corresponding to a column is crossed by the Hamiltonian path H, exactly when the row permutation π switches from a row with 0 to a row with 1 or vice versa. As all the ones are in 2 consecutive blocks, each such cut can be crossed at most four times. Introducing the dummy node corresponds to introducing a row with all zeros as the first row which clearly does not change 2-C1 property. Also, a Hamiltonian path crosses each singleton cut at most two times and cut C_{uv} at most two times for any $u, v \in V$. Hence, there exists a spanning tree which crosses every cut in $\mathcal{C}$ at most four times.

Now, for the other direction we show that if there exists a spanning tree T which cuts every cut in $\mathcal{C}$ at most 4 times then there exists that a permutation π which has the 2-C1 property. We claim that any such tree must be a Hamiltonian path. As each singleton vertex is a cut in $\mathcal{C}$, hence degree of each vertex is at most four. Suppose there exists a vertex u with degree four. For $n > 5$, there exists a vertex v which is not a neighbor of u. But, then the cut C_{uv} is crossed at least five times. Hence, all vertices have degree at most three. Suppose, for the sake of contradiction there exists a vertex u such that $deg_T(u) = 3$. Consider any vertex v which is not a neighbor of u. As T crosses C_{uv} at most four times, so $deg_T(v) = 1$. This implies that the total sum of degrees of nodes in T is at most $3 \cdot 4 + (n - 3)$. Since, the sum of degrees of vertices in a tree is $(2n - 2)$, we get $(2n - 2) \leq (n + 9)$ or $n \leq 11$, which is a contradiction assuming larger n. Hence, every vertex must have degree at most two in T showing that T is a Hamiltonian path.

Let the hamiltonian path be $(v_1, \ldots, v_k, s, v_{k+1}, v_n)$. Consider the following permutation of rows $(r_{k+1}, \ldots, r_n, r_1, \ldots, r_k)$ where v_i corresponds to row r_i in the transformation. We claim that in each column, there cannot be more than two blocks of ones. Suppose for the sake of contradiction, there exists a column c_i which has three blocks of ones. Thus, the cut corresponding to the Hamiltonian cycle formed by joining v_n

and v_1 must cross the cut corresponding to column c_i at least five times. But any cycle crosses any cut even number of times. Hence, it must cross the cut at least six times, but then the hamiltonian path must cross the cut at least five times, a contradiction. Hence, there exists a permutation which satisfies the 2-C1 property. This reduction shows that decision version of MCST problem for complete graphs is NP-complete. □

5 Conclusions

We have given two approximation algorithms for the MCST problem : a) a greedy algorithm which is combinatorial in nature and, b) an LP-based algorithm that uses randomized rounding in phases. To prove the guarantees for the greedy algorithm, we use a counting argument and we are unable to avoid a factor of r in the approximation. Recall that, r is the maximum number of cuts in the given family, that any edge e can cross. The LP-based randomized algorithm avoids the r factor in approximation and gives a spanning tree which has maximum crossing $O(\log n \cdot \text{OPT} + \log m)$ with high probability. It would be interesting to find a combinatorial algorithm with the same guarantees.

In the last section we showed that the MCST problem is NP-hard even for the case of complete graphs. However, the techniques used in this paper do not give any better guarantees for this special case. It would be interesting to see if better performance ratios can be obtained for the MCST problem on complete graphs.

References

1. N. Alon. A note on network reliability, *Discrete Probability and Algorithms, D. Aldous et al., eds., IMA Volumes in mathematics and its applicationsi*, Vol. 72, Springer Verlag (1995), 11-14.
2. K. Booth and G. Luker. Testing for the consecutive ones property, interval graphs and graph planarity using pq-tree algorithms. *Journal of Computer and System Sciences*, 13:335-379,1976.
3. Vittorio Bilo, Vineet Goyal, R. Ravi and Mohit Singh, To appear in *Proceedings of 7th. International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, 2004.
4. Vittorio Bilo and Michele Flammini. On the IP routing tables minimization with addresses reassignments. *In Proc. of the* 18^{th} *International Parallel and Distributed Processing Symposium(IPDPS).* IEEE Press, 2004. To appear.
5. Michele Flammini, Giorgio Gambosi and Stefano Salomone. Interval Routing schemes. *Algorithmica* 16(6):549-568,1996.
6. Michele Flammini, Alberto Marchetti-Spaccamela and Jan van Leeuwen. The Complexity of Interval Routing on Random Graphs. *The Computer Journal*, 41(1):16-25, 1998.
7. Paul W. Goldberg, C. Golumbic, Martin, Haim Kaplan, and Ron Shamir. Four Strikes against Physical Mapping. *Journal of Computational Biology*, 2(1):139-152, 1995.
8. David S. Greenberg and Sorin Istrail.Physical mapping by STS Hybridization: Algorithmic strategies and the challenges of software evaluation.*Journal of Computational Biology*, 2(2):219-273,1995.
9. Michael R. Garey and David S. Johnson *Computers and Intractibility: A guide to the Theory of NP-completeness.* W. H. Freeman and Company, New York, 1979.

10. M. Furer and B. Raghavachari. Approximating the minimum degree spanning tree to within one from the optimal degree. *In Proceedings of the Third Annual ACM–SIAM Symposium on Discrete Algorithms (SODA'92)*, pages 317–324, 1992.
11. Ajit Agrawal, Philip Klein, R. Ravi. When trees collide: An approximation algorithm for the generalized Steiner problem on networks. *Proceedings of the Twenty-third Annual ACM Symposium on Theory of Computing*, pp. 134–144, 1991.
12. P. Raghavan and C. Thompson. Randomized Rounding. In *Combinatorica*, volume 7, pages 365-374, 1987.