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,mohit}@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 then demonstrate that the problem remains NP-hard even when G is complete. For the latter case, we design a randomized algorithm that gives a tree T with crossing $O((\log m + \log n) \cdot (\text{OPT} + \log n))$ w.h.p., where OPT is the minimum crossing of any tree.

Our greedy analysis extends the traditional one used for set cover. The randomized algorithm rounds a LP relaxation of a corresponding subproblem in stages.

1 Introduction

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, where the crossing of a cut C_i is defined as $|E(T) \cap C_i|$. If the family of cuts is $\mathcal{C}$ = $\{(v, V \setminus v) : v \in V\}$, then the MCST problem reduces to finding the minimum degree spanning tree problem which has been widely studied [8]. Hence, NP-completeness of the minimum degree spanning tree problem [7] shows that MCST problem is NP-hard.

In this paper, we show approximation guarantees for the greedy algorithm for the MCST problem.

Theorem 1. *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.*

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 2. *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.*

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

A proof of the above theorem appears in the Appendix. The particular case of complete graphs finds application in fields as varied as IP routing and computational biology. We give improved algorithm for the MCST problem on complete graph which gives better performance guarantees.

Theorem 3. *There is a randomized LP rounding based algorithm, which given a complete 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 $O((\log m + \log n) \cdot (OPT + \log n))$, where OPT is the maximum crossing of an optimal tree.*

1.1 Motivation: Chimerism in Physical Mapping

The MCST problem finds important applications 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. 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 [1]. 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 [5].

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. Any 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 [2] 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 [2] it is shown that, given any r-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 $2r \log m$-approximation algorithm exists for MIN-IP, which exploits a matrix representation of the instances of the problem.

Similar applications can be found also in designing interval routing schemes as proposed in [3, 4].

1.4 Related Work

As observed earlier, the minimum degree spanning tree problem is a special case of the MCST problem. The best result for the minimum degree spanning tree problem are due to Furer and Raghavachari [8]. They construct a spanning tree with maximum degree at most $\Delta^* + 1$ where Δ^* is the maximum degree of the optimal tree. The $vMST$ problem has been considered by Greenberg and Istrail [6]. They give solution of cost $O(s(A) \cdot OPT + \log n)$. Here $s(A) = max_{1 \leq i \leq n} \sum_{j=1}^{n} a_{ij}$. Note that r in Theorem 1 is different from $s(A)$ in [6]: r is the maximum number of cuts a given edge e can cross, where the cuts are defined by columns of A; $s(A)$ is the sparsity of A i.e. the maximum number of 1's in any row in A. Observe that $r \leq 2 \cdot s(A)$, but $s(A)$ can be as bad as m. Hence, our algorithm gives comparable or better performance guarantee than the algorithm in [6].

The paper is organized as follows. In Section 2, we describe a greedy algorithm for the MCST problem and prove Theorem 1. In Section 3, we give a randomized algorithm for the special case and prove the guarantees of Theorem 3. In the Appendix, we show that the MCST problem is NP-hard even for complete graphs.

2 Greedy Algorithm for the General Case

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 $e \in G$ which join 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 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.

The proof of Theorem 1 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, 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,

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

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

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

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

$$\begin{aligned} n &\geq (1 + \tfrac{1}{2r*opt})^l \\ \Rightarrow \log n &\geq l \log(1 + \tfrac{1}{2r*opt}) \end{aligned}$$

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

$$\begin{aligned} \log n &\geq l(\tfrac{1}{2r*opt}(1 - \tfrac{1}{4r*opt})) \geq l \tfrac{1}{4r*opt} \\ \Rightarrow \quad l &\leq 4r \log n * opt \end{aligned}$$

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

3 A Randomized Algorithm for the Case of Complete Graphs

In this section, we describe a randomized algorithm for MCST for complete graphs and prove that it gives a tree with maximum crossing $O((\log m + \log n) \cdot (\text{OPT} + \log n))$ with high probability, where n is the number of vertices in G and m is the number of cuts in $\mathcal{C}$.

The idea is the following : Start with each vertex as a different component and merge components in phases until a connected subgraph is obtained. In a phase, each component is represented by an arbitrarily chosen vertex of the component. We carefully select some edges between the representative vertices by solving a multicommodity flow problem in each phase, so that the cuts in $\mathcal{C}$ are not crossed "too much". We ensure that at least one edge is chosen out of each representative in every phase. Hence, the number of components reduces by at least a factor of two and thus a connected subgraph is obtained in at most $\log_2 n$ phases.

In phase p, we solve the following multicommodity flow problem on a graph G' constructed from a complete graph G_p (on the representative vertices in this phase) as follows. Let $V(G_p) = \{v_1, v_2, \ldots, v_{n_p}\}$.

- For each undirected edge $e = (u, v)$, add two directed edges $e_f = (u, v)$ and $e_r = (v, u)$ in G',
- For each vertex $v_i \in V(G_p)$ introduce a new vertex s_{v_i} in $V(G')$ and
- $\forall v_j \in V(G_p), j \neq i$, add the directed edge (v_j, s_{v_i}) in G'.

Now, the flow problem on G' is the following. Each vertex $v_i \in V(G)$ is required to send a unit flow of commodity i to s_{v_i}. Let $f_1, f_2, \ldots, f_n$ be the flows associated with each of the n commodities. Let $f_i(v)$ denote the net flow of i^{th} commodity into the vertex v. The following integer program accomplishes our goal.

$$\begin{array}{lll}
\min & z & \\
\text{s.t.} \quad z & \geq \sum_{e \in E(G) \cap C} X_e & \forall C \in \mathcal{C} \\
\forall i = 1, \ldots, n_p & & \\
f_i(v) & = 0 & \forall v \in V(G') \setminus \{v_i, s_{v_i}\} \\
f_i(v_i) & = -1 & \\
f_i(s_{v_i}) & = 1 & \\
f_i(e) & \in \{0, 1\} & \forall e \in E(G')
\end{array}$$

We now describe the algorithm for the MCST problem. We will construct a connected subgraph H with a low *maximum crossing*.

1. Initialize $V(H) \leftarrow V(G)$, $E(H) \leftarrow \phi$, $G_0 \leftarrow G$, $R_0 \leftarrow V(G)$, $p \leftarrow 0$.
2. While H is not connected
   (a) Construct G' from G_p. Solve the LP-relaxation of the corresponding integer program for phase p and obtain an integral solution $\hat{X}$ by *randomized rounding* of the optimum LP solution [10].
   (b) Let $E' = \{e \in G_p : \hat{X}_e > 0\}$. $E(H) \leftarrow E(H) \cup E'$.
   (c) $p \leftarrow p + 1$. Let R_p be the set of representative vertices(chosen arbitrarily one for each connected component of H), G_p is the complete graph on the vertices of R_p.

Let T^* be a optimal tree for the MCST problem and let OPT be the maximum crossing of any cut in T^*.

Proposition 1. *Let z_p^* be the optimum to the LP-relaxation in phase p. Then $z_p^* \leq$ 2OPT.*

Proof. We can construct a feasible solution of the LP from the optimum tree T^* of value at most 2OPT. Let $R_i = \{v_1, \ldots, v_{n_p}\}$ be the set of representatives in phase i. From the Tree Pairing Lemma [9], there exists a matching M between vertices of R_i such that the paths in T^* between the matched pairs of vertices are edge disjoint. We can use this matching to construct a feasible solution to the LP. Send a unit flow of commodity i on the directed path $P_{v_i, v_j} \cup (v_j, s_{v_i})$ and of commodity j on the path $P_{v_j, v_i} \cup (v_i, s_{v_j})$, where $P(u, v)$ is the unique path in tree T^* between matched pairs u and v. The above flow is a feasible flow as it satisfies all the flow constraints of the LP. Every edge of T^* carries at most two units of flow. Hence, the objective value z for this feasible flow, is at most 2OPT. Therefore, $z_p^* \leq$ 2OPT.

Proposition 2. *If an edge $e = (u, v)$ crosses a cut C, then any other path between u and v also crosses the cut C at least once.*

Proof. If we remove all the edges in C from G, then u and v would be disconnected. Thus, every path from u to v contains an edge of C.

We will use Observation 2, to obtain a special kind of optimum solution such that each flow path uses only two edges. Consider the flow decomposition for commodity i in the optimum solution of the LP-relaxation and consider a flow path $P =< v_i, v_{i_1}, v_{i_2}, \ldots, v_{i_k}, s_{v_i} >$. We can replace P by the path $P' =< v_i, v_{i_k}, s_{v_i} >$ without increasing the maximum crossing. From Observation 2, we know that any cut that the edge (v_i, v_{i_k}) crosses will be crossed at least once by the path P. Therefore, P' only reduces the number of crossings for the cuts in $\mathcal{C}$ and so we can replace P by P'. Thus, we can obtain a fractional optimum solution S^* such that each flow path uses only two edges.

3.1 Rounding S^* to an integral solution

Let us describe the rounding of the fractional multicommodity flow obtained by solving the LP relaxation corresponding to phase p. The flow corresponding to each commodity is rounded independently of others. For each commodity $i = 1, \ldots, n_p$, choose an edge $e = (v_i, v_j)$ with probability $f_i(v_i, v_j)$. The corresponding flow is routed through the path $< v_i, v_j, s_{v_i} >$ and the edge (v_i, v_j) is included in the subgraph H. This is repeated for every commodity independently.

In phase p, let the fractional optimum flow be f^* and the optimum LP solution be z^*. Let $z(C)$ denote the number of edges crossing a cut $C \in \mathcal{C}$. Consider Y_j, a 0-1 variable associated with the j^{th} commodity, where

$$Y_j = \begin{cases} 1 \text{ if the integral flow crosses C} \\ 0 \text{ otherwise} \end{cases}$$

Therefore,

$$Pr(Y_j = 1) = \sum_{e \in E(G_p) \cap C} \tilde{f}_j(e)$$

$$z(C) = \sum_{j=1}^{n_i} Y_j$$

$$\begin{aligned} E[z(C)] &= \textstyle\sum_{j=1}^{n_i} \sum_{e \in E(G_i) \cap C} \tilde{f}_j(e) \\ &= \textstyle\sum_{e \in E(G_i) \cap C} \sum_{j=1}^{n_i} \tilde{f}_j(e) \\ &= \textstyle\sum_{e \in E(G_i) \cap C} X_e \\ &\leq \tilde{z} \quad \leq \quad 2 \cdot \mathrm{OPT} \end{aligned}$$

$z(C)$ is the sum of independent Bernoulli trials. Thus, we can use Chernoff bounds to bound the tail probability

$$Pr(|z(C) - E[z(C)]| > k\beta) \quad \leq \quad exp(-\frac{k^2\beta^2}{2E[z(C)]})$$

Let $\beta = E[z(C)] + \log n$ and $k = \log_n m + 2$[3]. Therefore,

$$\begin{aligned} Pr(|z(C) - E[z(C)]| > k\beta) &\leq exp(-\tfrac{k^2(E[z(C)]+\log n)}{2}) \\ &< exp(-\tfrac{(2\log_n m+4)\log n}{2}) \\ &= \tfrac{1}{mn^2} \end{aligned}$$

Since $E[z(C)] \leq 2\text{OPT}$, we have that $Pr(z(C) > (2(k+1)\text{OPT}+k\log n)) < \frac{1}{mn^2}$ or $Pr(z(C) > O((\log_n m + 3) \cdot OPT + (\log_n m + 2) \cdot \log n)) < \frac{1}{mn^2}$ for any cut $C \in \mathcal{C}$ in any phase p. We say that a "bad" event occurs in a phase p if some cut $C \in \mathcal{C}$ has a high crossing in that phase. Thus, from the union bound we have $Pr(\text{bad event occurs in phase p}) < \frac{1}{n^2}$. The algorithm has at most $\log_2 n$ phases. Thus,

$$Pr(\text{"bad" event occurs in any phase}) < \frac{\log n}{n^2} \tag{4}$$

Thus, we have shown that in every phase the crossing of every cut is $O((\log_n m + 3)\text{OPT} + (\log_n m + 2) \cdot \log n)$ with high probability. Hence, we obtain a solution of maximum crossing $O((\log_2 m + \log_2 n) \cdot (\text{OPT} + \log_2 n))$ with probability at least $(1 - \frac{\log n}{n^2})$. □

4 Future Work

We believe that better performance ratios can be obtained particulary for the MCST problem on complete graphs. Furthermore, more sophisticated methods than a simple greedy approach should be able to remove the factor of r in the general case.

References

1. 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.*
2. 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.
3. Michele Flammini, Giorgio Gambosi and Stefano Salomone. Interval Routing schemes. *Algorithmica 16(6):549-568,1996.*
4. 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.*
5. 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.*
6. 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.*

[3] For $m \geq n$, setting $k = \sqrt{(log_n m + 2)}$ gives a slightly better solution with maximum crossing $O(\sqrt{\log m \log n}(OPT + \log n))$

7. 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.
8. 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.*
9. 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.*
10. P. Raghavan and C. Thompson. Randomized Rounding. *In Combinatorica, volume 7, pages 365-374, 1987.*

Appendix: 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 [6].
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 with rows with 1 form one side of the cut. The dummy vertex s is always on the 0-side of each 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 with a row with 1 or vice versa. As all the ones are in 2 consecutive blocks, each 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-C1P 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 $3 * 4 + (n - 3)$. Hence, $2n - 2 \leq n + 9$ or equivalently, $n \leq 11$ 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. $\square$