Multiagent MST Cover: Pleasing All Optimally via a Simple Voting Rule

Bo Li [1], **Xiaowei Wu** [2], **Chenyang Xu** [3,4*], **Ruilong Zhang** [5]

[1] Department of Computing, The Hong Kong Polytechnic University
[2] IOTSC, University of Macau
[3] Software Engineering Institute, East China Normal University
[4] College of Computer Science, Zhejiang University
[5] Department of Computer Science, City University of Hong Kong
comp-bo.li@polyu.edu.hk, xiaoweiwu@um.edu.mo, xcy1995@zju.edu.cn, ruilzhang4-c@my.cityu.edu.hk

Abstract

Given a connected graph on whose edges we can build roads to connect the nodes, a number of agents hold possibly different perspectives on which edges should be selected by assigning different edge weights. Our task is to build a minimum number of roads so that every agent has a spanning tree in the built subgraph whose weight is the same as a minimum spanning tree in the original graph. We first show that this problem is NP-hard and does not admit better than $((1 - o(1)) \ln k)$-approximation polynomial-time algorithms unless $\mathsf{P} = \mathsf{NP}$, where k is the number of agents. We then give a simple voting algorithm with an optimal approximation ratio. Moreover, our algorithm only needs to access the agents' rankings on the edges. Finally, we extend our results to submodular objective functions and Matroid rank constraints.

1 Introduction

Minimum spanning tree (MST) is one of the most fundamental problems in graph theory, whose objective is to find a subgraph in a given graph to span all the nodes with minimum total weight. It finds wide applications in computer science, among which a typical scenario is network construction, such as fiber optic network design (Bachhiesl et al. 2002), highway system design (Ahuja, Magnanti, and Orlin 1993), FIR filter implementation (Ohlsson, Gustafsson, and Wanhammar 2004), and so on. The problem becomes trickier when the system interacts with multiple heterogeneous entities who hold possibly different perspectives on measuring the weights of the edges in the graph. For example, when a government is planning the traffic in a city, many bureaus, such as water resources, communications, construction, finance, and information industry, are involved, but they may have different concerns and thus hold different opinions on which roads should be built. Since there may not exist one MST that satisfies all bureaus, we focus on the minimum cost maximum social welfare problem by building the minimum number of roads so that all bureaus are satisfied. Formally, we are interested in the following optimization problem: Given a graph where different agents may have different weight functions on the edges, how can we select a minimum number of edges so that the induced subgraph contains at least one MST of the original graph for every agent.

Our problem is closely related to the multiple objective minimum spanning tree (MOST) problem, whose goal is to select a spanning tree T in the graph to minimize $(g_1(T), \ldots, g_k(T))$, where each $g_i(\cdot)$ represents a different weight function. Various multi-dimensional optimality concepts have been investigated in the literature (Hassin and Levin 2004; Hong, Chung, and Park 2004; Majumder et al. 2022; Ruzika and Hamacher 2009), among which two of the most popular ones are min-sum (Correia, Paquete, and Figueira 2021; Fernandes et al. 2019) and min-max (Darmann, Klamler, and Pferschy 2009; Escoffier, Gourvès, and Monnot 2013), i.e., finding a spanning tree T to minimize $\sum_{i\in[k]} g_i(T)$ or $\max_{i\in[k]} g_i(T)$. Although computing a minimum spanning tree in the single-objective setting is polynomial-time solvable, it appears to be NP-hard for both multi-dimensional optimality concepts. Similar problems are also studied in computation social choice, where the objective is to find one spanning tree such that the satisfaction of the least satisfied agent gets maximized under different "satisfaction" notions (Darmann 2016; Darmann, Klamler, and Pferschy 2009; Escoffier, Gourvès, and Monnot 2013; Gourvès, Monnot, and Tlilane 2015). The key difference between the existing research and our work is that we are not targeting a spanning tree; instead, we are allowed to select a spanning *subgraph* and aim to please all agents with a minimum number of edges.

Regardless of which multi-dimensional optimality concept is adopted, targeting a spanning tree may lead to extremely bad situations for some agents. We show an example in Fig. 1, where ∞ denotes a large constant. There are two agents. The unique MST of agent 1 and agent 2 is $\{(a,b),(a,c),(c,d)\}$ and $\{(a,b),(a,d),(c,d)\}$, respectively. If we restrict any feasible solution to be a spanning tree, no matter which spanning tree we select, there will always be half of the agents that dislike it very much. For example, the selected spanning tree is $T = \{(a,b),(a,d),(c,d)\}$ in the figure. Then agent 1 is de-

*All authors (ordered alphabetically) have equal contributions and are corresponding authors. The work was done when Chenyang Xu was a student at Zhejiang University and Ruilong Zhang visited University of Macau.

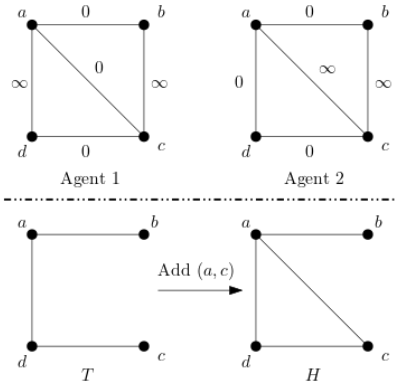

Figure 1: An illustration of the satisfiability issue when the solution is restricted to a spanning tree.

pressed as her valuation of the selected spanning tree is ∞, much larger than her original MST's value 0. But if we are allowed to add one more edge to the solution, all agents will be satisfied because each of they can find her original MST from the given subgraph $H = \{ (a, b), (a, c), (a, d), (c, d) \}$.

In the urban traffic planning example, and more practical applications, building a spanning tree is more economical than building a spanning subgraph with more edges. However, it may make some agents unsatisfied and thus, lead to poor social welfare. On top of maximizing social welfare, we want to find an economical way to please all agents, which motivates our model.

1.1 Our Contributions

We study the multiagent MST cover problem (MMCP), where the goal is to select a minimum subgraph that covers a minimum spanning tree of the original graph for every agent. In a nutshell, our main results can be summarized as follows.

Main Results. We show that the multiagent MST cover problem is as hard as set cover and design an optimal logarithmic approximation algorithm, which also works when the algorithm only accesses the ordinal information of agent preferences. Further, the proposed algorithm can be extended to submodular objective functions and Matroid rank constraints.

In a bit more details, we first show that the optimization problem is NP-hard via a reduction from the set cover problem, and does not admit better than $((1 - o(1)) \ln k)$-approximate polynomial-time algorithms unless $\mathsf{P} = \mathsf{NP}$, where k is the number of agents. Interestingly, a special case of the decision problem—deciding whether there is a spanning tree that is minimum for all agents, can be solved in polynomial time.

We then design a polynomial-time algorithm that achieves the optimal approximation ratio. Intuitively, in each round of our algorithm, we ask the agents which edge should be added to the solution and each agent votes for the ones that make the solution "closer" to one of her minimum spanning trees. The algorithm adds the edge that receives the most votes and stops when all agents find a minimum spanning tree in the solution. Technically, we define a progress function to evaluate how close our partial solution is to satisfying all agents, and then show that our algorithm is equivalent to adding edges to maximize the marginal progress. By proving that the progress function is submodular, we obtain the logarithmic approximation.

Our algorithm is robust in two senses. First, it does not need to access the cardinal edge-weights of the agents; as long as we have the ordinal rankings from the agents, the approximation ratio is guaranteed. This makes our algorithm practically useful as it may be hard for the government to gather all the accurate numerical values from the agents. Second, our algorithm can be extended to more complicated scenarios. For example, due to the heterogeneity of the edges (e.g., lengths and geologic structures) the construction cost of different edges can be different for the government. Thus instead of selecting the minimum number of edges, the government wants to minimize the total construction cost of the selected edges. We show that our algorithmic idea still works in the case that the construction cost function (over edges) is monotone and submodular. In addition to the generalized objective, we also extend the minimum spanning tree constraints to maximum Matroids rank, where our results also apply. Due to space limit, the results of the generalized models are omitted in this version and can be found in the full version (Li et al. 2022).

1.2 Other Related Works

Multi-objective Spanning Tree. Finding a minimum spanning tree is a fundamental problem in multi-objective optimization (Ehrgott 2005). See (Ruzika and Hamacher 2009) for a survey. Typically, we are given a connected undirected graph $G(V, E)$. Each edge $e \in E$ has a positive weight vector $\mathbf{w}(e) = (w_1(e), \ldots, w_k(e))$. For every spanning tree T, we have a weight vector $\mathbf{w}(T) = (w_i(T))_{i \in [k]}$ with $w_i(T) = \sum_{e \in E} w_i(e)$. The goal is to compute a spanning tree T such that T is Pareto optimal. This problem has been shown to be NP-hard even if the weight vector only has two coordinates (Hamacher and Ruhe 1994) in which the problem is called the bi-objective spanning tree minimization problem. The literature mainly looks into the approximation algorithms and exact exponential time algorithms. Bi-objective spanning tree minimization has also been studied extensively (Amorosi and Puerto 2022; de Sousa, Santos, and Aloise 2015). For approximation algorithms, a fully polynomial-time approximation scheme (FPTAS) is proposed by (Papadimitriou and Yannakakis 2000). For exact algorithms, most existing algorithms are widely used evolutionary methods in multi-objective optimization (Lai and Xia 2019; Prakash, Patvardhan, and Srivastav 2020).

Fair Allocation with Public Goods. Another related model of our problem is fair allocation with public goods. In the model, we aim to select a subset of given public goods such that the agents' fairness objective is optimized. Conitzer, Freeman, and Shah (2017) first introduce this model. They aim to select goods to satisfy the fairness notion of propositional share. Then Fain, Munagala, and

Shah (2018) propose a new fairness notion for public goods which generalizes the propositional share. The authors also consider some constraints of goods, i.e., the selected goods must satisfy some properties, like matching, packing, and matroid. Recently, Fluschnik et al. (2019) and Garg, Kulkarni, and Murhekar (2021) set Nash social welfare as the fairness notion and aim to select goods such that the Nash social welfare is maximized.

Partial Information Setting. In the current work, due to the practical concern when it is hard for the decision maker to elicit agents' complete information of their preferences, we assume that the algorithm only has access to their ordinal rankings. Similar settings have been widely investigated in computational social choice. Procaccia and Rosenschein (2006) consider elections under the Borda voting rule, and introduces *distortion* which is used to bound the worst-case deterioration of the optimal objective with limited access of preferences. Caragiannis and Procaccia (2011) and Caragiannis et al. (2017) continue to study the same problem under the Plurality rule. There is also a line of works, e.g., (Anshelevich et al. 2018; Munagala and Wang 2019; Gkatzelis, Halpern, and Shah 2020), that studies the metric social choice with partial information when the alternatives are points in a metric space. We refer the readers to (Anshelevich et al. 2021) for a detailed survey.

1.3 Roadmap

The formal definition of multiagent MST cover is stated in Section 2. Section 3 proves that the problem is set-cover hard, while Section 4 gives a simple and efficient algorithm to determine whether there is a spanning tree that is an MST for all agents. In Section 5, we present our optimal logarithmic approximation algorithm. Finally, we conclude this paper and discuss some interesting future works in Section 6.

2 Preliminaries

Consider a connected undirected graph $G(V, E)$ with n nodes and m edges. There is a set K of k agents, where each agent $i \in K$ has a preference $\preceq_i$ over the edges E. Denote by $\preceq = (\preceq_1, \ldots, \preceq_k)$ the preference profile of all agents. For any two edges $a, b \in E$, $a \prec_i b$[1] means that from the view of agent i, edge a is cheaper than edge b, while $a \sim_i b$ represents the case that agent i is indifferent between a and b. Clearly, $\preceq_i$ can be denoted by κ_i equivalence classes $\mathcal{E}(\preceq_i) := (E_i^1, \ldots, E_i^{\kappa_i})$ (in increasing order of costs), where each E_i^j contains the edges among which agent i is indifferent, and κ_i is the number of her equivalence classes. Accordingly, for any two edges $a \in E_i^j$ and $b \in E_i^\ell$, $a \sim_i b$ implies $j = \ell$ and $a \prec_i b$ implies $j < \ell$.

Agents may have arbitrary cardinal weight function on the edges as long as they are *consistent* with $\preceq$. Specifically, agent $i \in K$ has weight function $w_i : E \to \mathbb{R}_{\geq 0}$ that is consistent with $\preceq_i$ if w_i satisfies: $w_i(a) < w_i(b)$ (resp. $w_i(a) = w_i(b)$) if and only if $a \prec_i b$ (resp. $a \sim_i b$) for any $a, b \in E$. We denote by $\mathbf{w} = (w_1, \ldots, w_k)$ the weight profile of all agents. In our model, the algorithms only have access to the preferences $\preceq$, but not the weight functions $\mathbf{w}$.

[1]Since agents prefer smaller cost in our model, we use $a \prec_i b$ to denote that $w_i(a) < w_i(b)$, which might be different from other literature of computational social choice, in which $a \prec_i b$ means that agent i prefers b than a.

Algorithm 1: Kruskal's Algorithm

Input: A connected undirected graph $G(V, E)$; a preference $\preceq$ over E.
Output: An MST T of G under $\preceq$.

```
T ← ∅; i ← 1.
Let (e_1, e_2, ..., e_m) be the list of edges sorted by ⪯ (break ties arbitrarily).
while i ≤ m do
    if there is no cycle in T ∪ {e_i} then
        T ← T ∪ {e_i}.
        i ← i + 1.
    end if
end while
return T.
```

Given any graph $G(V, E)$ and a set of edges $S \subseteq E$, let $G[S] := (V, S)$ be the subgraph of G induced by S. Under the weight function w, a *minimum spanning tree* (MST) of G is an induced subgraph $G[S]$ that is connected, and $w(S) := \sum_{e \in S} w(e)$ is minimized. When the context is clear, we also use S to denote the induced subgraph $G[S]$. Let $\mathsf{MST}(G, w)$ and $\mathsf{MST}(G, \preceq)$ be the set of all MSTs of G under weight function w and preference profile $\preceq$, respectively, where a spanning tree T is minimum under $\preceq$ if T is an MST under any weight functions that are consistent with $\preceq$. The following lemma captures the equivalence between $\mathsf{MST}(G, \preceq)$ and $\mathsf{MST}(G, w)$.

Lemma 1. *For any preference profile $\preceq$ and weight function w consistent with it,* $\mathsf{MST}(G, w) = \mathsf{MST}(G, \preceq)$.

The proof of Lemma 1 is omitted and can be found in the full version (Li et al. 2022). For simplicity, denote $\mathsf{MST}(G, \preceq_i)$ by MST_i for each agent $i \in K$. For a positive integer x, we use $[x]$ to denote $\{1, 2, \ldots, x\}$. Based on Lemma 1, checking whether a spanning tree is an MST under some preference $\preceq$ can be done in polynomial time.

A feasible solution of the *multiagent MST cover problem* (MMCP) is a set of edges H such that for any agent i, there exists $T \subseteq H$ which is an MST for agent i. Namely, any agent has at least one MST that is covered by H. Call such an edge set an *MST cover* of all agents. Our goal is to find an MST cover for all agents with the minimum size. Note that in the full version (Li et al. 2022), we also consider the weighted version of MMCP, in which we have a cost function $c : 2^E \to \mathbb{R}_{\geq 0}$ on the edges, and the objective is to find an MST cover H of all agents with minimum cost $c(H)$.

By Lemma 1, given any connected undirected graph $G(V, E)$ with a preference $\preceq$ over E, it is well-known that Kruskal's algorithm (Kruskal 1956) computes an MST of G. For completeness, we restate the algorithm in Algorithm 1. Let $\mathsf{Kruskal}(G, \preceq)$ be the returned MST by Kruskal's algorithm given graph G and preference $\preceq$.

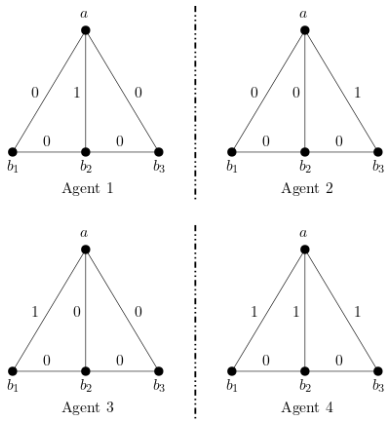

Figure 2: An illustration for the reduction. The set cover instance (S, U) is as follows: $U = \{ e_1, e_2, e_3 \}$ and $S = \{ s_1, s_2, s_3 \}$ with $s_1 = \{ e_1, e_2 \}$, $s_2 = \{ e_2, e_3 \}$ and $s_3 = \{ e_1, e_3 \}$. We construct a graph $G(V, E)$ with $V = \{ a, b_1, b_2, b_3 \}$ and $E = \{ (a, b_1), (a, b_2), (a, b_3) \} \cup \{ (b_1, b_2), (b_2, b_3) \}$. There are four agents with distinct weight vectors shown in the figure.

3 Hardness Results

In this section, we show that MMCP is NP-hard even if (i) all weight functions $w_1, \ldots, w_k$ are given; (ii) all weight functions are binary, i.e., for any $i \in K$ and $e \in E$, $w_i(e) \in \{ 0, 1 \}$. We refer to this special case as *MMCP with binary weights*.

Theorem 1. *MMCP with binary weights is* NP*-hard.*

Proof. The NP hardness is proved by a reduction from Set Cover (SC) to MMCP in which $w_i(e) \in \{ 0, 1 \}$ for any $i \in K$ and $e \in E$. In the set cover problem, we are given a ground element set $U = \{ e_1, \ldots, e_p \}$ and a subset collection $S = \{ s_1, \ldots, s_q \} \subseteq 2^U$. The goal is to find a minimum number of subsets in S whose union covers U.

Considering an arbitrary SC instance with p elements and q subsets, we construct an MMCP instance with a graph $G(V, E)$ and $p + 1$ agents as follows. The graph $G(V, E)$ contains $q + 1$ nodes and $2q - 1$ edges. Let $V = \{ a, b_1, \ldots, b_q \}$ where a is called an *up-node* and the nodes in $V \setminus \{ a \}$ are called *down-nodes*. For each $i \in [q]$, we connect a and b_i, and refer to edge (a, b_i) as an *up-edge*. For each $i \in [q - 1]$, connect b_i and b_{i+1}, and call this edge a *down-edge*. Now we define the weight functions of the $(p + 1)$ agents. First, let all the down-edges have weight 0 for all agents. Then, for each agent $j \in [p]$, let $w_j(a, b_i) = 0$ if element $e_j \in s_i$; $w_j(a, b_i) = 1$ otherwise. Note that by this construction, the value of MST for every agent $j \in [p]$ is 0. Let the last agent have weight 1 on all up-edges, and thus the MST's value for agent $p + 1$ is 1. It is easy to see that the reduction can be implemented in polynomial time. See Fig. 2 for an illustration.

Observe that any MST cover H of agents must contain all down-edges; otherwise, H does not cover any MST of the last agent. Moreover, H must contain at least one up-edge e with $w_j(e) = 0$ for any $j \in [p]$ due to a similar argument.

Algorithm 2: Computing a Perfect MST Cover

Input: A connected undirected graph $G(V, E)$; Agent set K and preferences $\preceq = (\preceq_1, \ldots, \preceq_k)$.
Output: A spanning tree T of G.

```
Sort all edges in E by the rank vector σ in lexicographical order.
Let ≼' be the ordering of the edges after sorting.
T ← Kruskal(G, ≼').
return T.
```

By interpreting picking edge (a, b_i) in MMCP as selecting set s_i, there is a one-to-one correspondence between feasible solutions to these two problems. Thus, computing the minimum MST cover H^* corresponds to finding an optimal solution for the set cover problem, and the hardness result follows. □

Building on the reduction above, we can further show an inapproximability bound for MMCP with binary weights. We state the theorem in the following and defer the proof to the full version (Li et al. 2022).

Theorem 2. *Unless* P = NP*, any polynomial-time algorithm for MMCP with binary weights has approximation ratio at least* $(1 - o(1)) \ln k$*, where* k *is the number of agents.*

4 Perfect MST Cover

Although MMCP is NP-hard and has an inapproximability of $(1 - o(1)) \cdot \ln k$, we find that deciding whether there is a spanning tree that is an MST for all agents can be decided efficiently. We refer to this common MST of all agents as a *Perfect MST Cover*. This section shows a polynomial time algorithm to decide the existence of a perfect MST cover.

Given any agent i with her preference $\preceq_i$, recall that $\mathcal{E}(\preceq_i) = (E_i^1, \ldots, E_i^{\kappa_i})$ is an ordered partition of E generated by $\preceq_i$. Let $\sigma_i : E \to [m]$ be a rank function for agent i, where $\sigma_i(e) = j$ if $e \in E_i^j$, i.e., $\sigma_i(e)$ is the rank of edge e for agent i. Fix an arbitrary ordering of agents and let $\boldsymbol{\sigma}(e) = (\sigma_1(e), \ldots, \sigma_k(e)) \in [m]^k$ be the rank vector of edge e. Note that $\boldsymbol{\sigma}$ induces a lexicographical ordering on the edges, i.e., $\boldsymbol{\sigma}(e_1) < \boldsymbol{\sigma}(e_2)$ if and only if there exists $i \in K$ such that $\sigma_j(e_1) = \sigma_j(e_2)$ for all $j < i$ and $\sigma_i(e_1) < \sigma_i(e_2)$.

The description of the algorithm is given in Algorithm 2. Clearly, Algorithm 2 is a polynomial time algorithm because sorting all edges by the rank vector $\boldsymbol{\sigma}$ takes $\mathcal{O}(km \log m)$ time by aggregating the preferences, and Kruskal's Algorithm runs in $\mathcal{O}(m \log n)$. The following theorem shows that if an instance admits a perfect MST cover, then Algorithm 2 computes one.

Theorem 3. *Given any instance of MMCP, Algorithm 2 returns a perfect MST cover* T *if and only if the instance admits one.*

Proof. Suppose the instance admits a perfect MST cover. Consider any weight function $\mathbf{w} = (w_1, \ldots, w_k)$ that is consistent with the given preference $\preceq = (\preceq_1, \ldots, \preceq_k)$. We show that T is a perfect MST cover under the weight

function $\mathbf{w}$. Let $w_i(T)$ be the total weight of T for agent $i \in K$. According to the statement of Algorithm 2, T is an MST under the preference $\preceq'$. Thus, among all spanning trees of the graph, T has the minimum weight vector $(w_1(T), \ldots, w_k(T))$ (lexicographical order). So if there exists a perfect MST cover T', we have

$$(w_1(T'), \ldots, w_k(T')) \geq (w_1(T), \ldots, w_k(T)).$$

On the other hand, due to the definition of a perfect MST cover, for any $i \in K$, we have

$$w_i(T) \geq w_i(T').$$

Thus, for all $i \in K$, we have $w_i(T) = w_i(T')$, implying that T is a perfect MST cover. □

Remark 1. *It deserves to note that there are other ways to compute a perfect MST cover besides implementing Algorithm 2. For example, we can arbitrarily assign each agent a weight function $w_i(\cdot)$ that is consistent with her preference, and then sum up all agents' weights $\bar{w}(\cdot) = \sum_i w_i(\cdot)$. We can compute an MST on $\bar{w}(\cdot)$ (denoted by T), and prove that there is a common MST if and only if $T \in \mathsf{MST}_i$ for any agent i.*

5 Multi-Round Plural Voting Algorithm

This section gives an $\mathcal{O}(\ln k)$-approximation algorithm (Algorithm 3) and shows the following main result (Theorem 4). Note that the approximation ratio is asymptotically optimal by the hardness result we have shown in Section 3.

Theorem 4. *For every instance of MMCP, Algorithm 3 computes an $\mathcal{O}(\ln k)$-approximation solution in polynomial time, where k is the number of agents.*

Algorithmic Framework. Our algorithm runs in rounds and adds exactly one edge to the solution H (which is initially empty) in each round until the solution is feasible, i.e., it contains an MST for every agent. At the beginning of each round, every agent specifies a candidate edge set $S_i \subseteq E \setminus H$ from the unselected edges. Intuitively, the candidate edges in S_i are those agent i prefers to select, given the current partial solution H. Then, our algorithm chooses the edge that appears in most candidate edge sets (break ties arbitrarily), e.g., the edge that receives the most "votes", and adds it to the solution set H. In other words, the algorithm repeats the following steps until H becomes feasible:

- each agent $i \in K$ specifies a candidate edge set $S_i \subseteq E \setminus H$;
- let $e^* \in \arg\max_{e \in E \setminus H} |\{i \in K : e \in S_i\}|$ be an edge that receives the most votes;
- add e to the solution: $H \leftarrow H \cup \{e\}$.

Apparently, the crucial part of the algorithm is how to define the candidate edge set for each agent in each round. In our algorithm, we introduce a *progress function* $f_i : 2^E \to \{0, 1, \ldots, n-1\}$ for each agent $i \in K$ that measures how much progress the current solution H has made for her, e.g., the larger $f_i(H)$ is, the more progress has been made for agent i. The candidate edges for agent i are naturally those whose inclusion to H makes $f_i(H)$ larger.

Algorithm 3: Multi-Round Plural Voting Algorithm

Input: A connected undirected graph $G(V, E)$; Agent set K with preferences $\preceq = (\preceq_1, \ldots, \preceq_k)$.
Output: A subgraph $H \subseteq E$.

```
H ← ∅.
while F(H) < k(n − 1) do
    e* ← arg max_{e∈E\H} {F(H ∪ {e}) − F(H)}.
    H ← H ∪ {e*}.
end while
return H.
```

Definition 1 (Progress Function)**.** *Given any agent $i \in K$ and any edge set $H \subseteq E$, the progress function f_i of agent i is defined as $f_i(H) := \max_{T \in \mathsf{MST}_i} \{|H \cap T|\}$.*

In other words, $f_i(H)$ is the maximum number of edges from H that can simultaneously appear in some MST of agent i. Note that the function f_i is monotone and $f_i(H) = n-1$ if and only if the set H contains an MST for agent i. Therefore, $(n-1) - f_i(H)$ indicates the minimum number of edges that need to be added into H so that the solution becomes feasible to agent i. The closer $f_i(H)$ is to $n-1$, the agent is more satisfied with the partial solution H.

Example. Let us consider the example from Fig. 1. Note that the unique minimum spanning tree for agent 1 and 2 are $T_1 = \{(a,b),(a,c),(c,d)\}$ and $T_2 = \{(a,b),(a,d),(c,d)\}$, respectively. If the partial solution $H = \{(a,b),(a,c)\}$, then $f_1(H) = 2$ (resp. $f_2(H) = 1$) since $|T_1 \cap H| = 2$ (resp. $|T_2 \cap H| = 1$). If we add one more edge (a,d) to H, the new partial solution becomes $H' = \{(a,b),(a,c),(a,d)\}$. In this case, we have $f_1(H') = 2$ and $f_2(H') = 2$. Since $f_2(H \cup \{(a,d)\}) > f_2(H)$, edge (a,d) is among the candidate edges of agent 2, given the partial solution H.

5.1 The Complete Algorithm

We now present the complete algorithm by implementing the definitions into the algorithm framework. Given the partial solution H, the candidate edge set of agent i is defined as

$$S_i := \{e \in E \setminus H : f_i(H \cup \{e\}) > f_i(H)\}.$$

Let function F be the sum of all agents' progress functions, i.e., $F(H) = \sum_{i \in K} f_i(H)$. Note that $F(H) \in \{0, 1, \ldots, k(n-1)\}$, and H is a feasible solution if and only if $F(H) = k(n-1)$. Since the functions $(f_i)_{i \in K}$ are monotone, so is F. Therefore, $k(n-1) - F(H)$ measures how far an edge set H is away from being feasible. Selecting the edge that appears in most candidate edge sets can be equivalently described as picking $e \in E \setminus H$ with maximum *marginal contribution* $F(H \cup \{e\}) - F(H)$ to the current partial solution H. The formal description of the algorithm can be found in Algorithm 3.

The analysis to show that Algorithm 3 computes an $\mathcal{O}(\ln k)$-approximation solution to MMCP in a polynomial time consists of the following three steps.

1. We show that given any partial solution $H \subseteq E$, the value of $F(H)$ can be computed in polynomial time. Since $F(H) = \sum_{i \in K} f_i(H)$, it suffices to provide an oracle for each agent i that answers the value of $f_i(H)$ for each queried set $H \subseteq E$. The following lemma is proved in Section 5.2.

   Lemma 2. *Given the preference $\preceq_i$ of agent i and a subset of edges $H \subseteq E$, $f_i(H)$ can be computed in polynomial time.*

2. We show that the function F is submodular. Again, since $F = \sum_{i \in K} f_i$, it suffices to show the submodularity of each function f_i. The following lemma is proved in Section 5.3.

   Lemma 3. *For all agent $i \in K$, the progress function f_i is submodular.*

3. We show that Algorithm 3 returns a solution H with $|H| \leq \mathcal{O}(\ln k) \cdot |H^*|$. Given the monotonicity and submodularity of F, we borrow a result in (Wolsey 1982) to prove Theorem 4.

   Lemma 4 ((Wolsey 1982))**.** *Given any integer-valued monotone submodular function $F : 2^E \to \mathbb{Z}$ with $F(\varnothing) = 0$, the algorithm that repeatedly includes the element with maximum marginal value computes a set H such that (where $d = \max_{e \in E}\{F(e)\}$)*

$$|H| \leq \left(\sum_{i=1}^{d} \frac{1}{i}\right) \cdot \min\{|S| : F(S) = F(E), S \subseteq E\}.$$

Given the above three lemmas, we prove Theorem 4.

Proof of Theorem 4: Regarding the correctness of the algorithm, since the returned set H satisfies $F(H) = k(n-1)$, the solution H is feasible. By definition of the progress function (Definition 1), f_i is integer-valued, monotone and $f_i(\varnothing) = 0$ for all $i \in K$. By Lemma 3, f_i is submodular. Therefore the function $F = \sum_{i \in K} f_i$ is integer-valued, monotone, submodular and $F(\varnothing) = 0$. Moreover, since $f_i(e) \leq 1$ for any agent i, we have $\max_{e \in E}\{F(e)\} \leq k$. Hence by Lemma 4, we have

$$|H| \leq \left(\sum_{i=1}^{k} \frac{1}{i}\right) \cdot |H^*| < (\ln k + 1) \cdot |H^*|.$$

Regarding the algorithm's running time, it suffices to show that there are a polynomial number of while-loops since $F(H)$ can be computed in polynomial time by Lemma 2 . As F is submodular and integer-valued, when $F(H) < k(n-1)$, there must exists e such that $F(H \cup \{e\}) - F(H) \geq 1$. Hence in each round, the element e^* selected in line 3 of Algorithm 3 increases $F(H)$ by at least one. Consequently, there are at most $k(n-1)$ rounds, and the algorithm runs in polynomial time. ■

5.2 Oracle for the Progress Function

In this subsection, we prove Lemma 2 by giving an algorithm (Algorithm 4) for the computation of $f_i(H)$.

Algorithm 4: Degrade Algorithm

Input: A connected undirected graph $G(V, E)$; Agent i with preference $\preceq_i$; An edge set $H \subseteq E$.
Output: The value of $f_i(H)$.
1: Let $\mathcal{E}_i \leftarrow (E_i^1, \ldots, E_i^{\kappa_i})$ be the ordered partition generated by the preference $\preceq_i$.
2: Let $\mathcal{E}_i' \leftarrow (E_i^1 \cap H, E_i^1 \setminus H, \ldots, E_i^{\kappa_i} \cap H, E_i^{\kappa_i} \setminus H)$ and remove empty equivalence classes.
3: Let $\preceq'$ be the preference such that $\mathcal{E}(\preceq') = \mathcal{E}_i'$.
4: $T \leftarrow \mathsf{Kruskal}(G, \preceq')$
5: $f_i(H) \leftarrow |T \cap H|$.
6: **return** $f_i(H)$.

Recall that agent i's preference $\preceq_i$ generates an ordered partition $\mathcal{E}(\preceq_i) = (E_i^1, \ldots, E_i^{\kappa_i})$ of the edge set E. Given any agent i and any edge set H, Algorithm 4 aims to find an MST T under $\preceq_i$ such that T uses a maximum number of edges from H. To do that, we use Kruskal's algorithm and break ties in favor of H. In other words, when running Kruskal's algorithm, within each equivalence class E_i^j, Algorithm 4 gives a higher priority to edges in $E_i^j \cap H$.

Proof of Lemma 2: We show that Algorithm 4 (which clearly runs in polynomial time) computes an MST $T \in \mathsf{MST}(G, \preceq_i)$ that has maximum overlap with H in line 4, which by definition implies $f_i(H) = |T \cap H|$, and proves Lemma 2. Note that $T \in \mathsf{MST}(G, \preceq_i)$ holds because Algorithm 4 is indeed Kruskal's algorithm (Algorithm 1), but with a carefully designed way to break ties.

Let $|T \cap H| = t$. For the sake of contradiction, suppose there exists another MST $T' \in \mathsf{MST}(G, \preceq_i)$ such that $|T' \cap H| \geq t + 1$. We construct a weight function w under which both T and T' are MSTs, but with different total weights, which is a contradiction. Fix any weight function w_i that is consistent with $\preceq_i$. We define another weight function w as follows, where $\epsilon > 0$ is arbitrarily small, e.g., $\epsilon \leq 1/m \cdot |w_i(R) - w_i(R')|$ for any $R, R' \subseteq E$ with $w_i(R) \neq w_i(R')$.

$$w(e) = \begin{cases} w_i(e), & e \notin H, \\ w_i(e) - \epsilon, & e \in H. \end{cases}$$

Observe that w is consistent with $\preceq_i'$, as we give edges in H a (very slightly) higher priority. Thus by line 4 of Algorithm 4 and Lemma 1, we have $T \in \mathsf{MST}(G, \preceq_i') = \mathsf{MST}(G, w)$. On the other hand, since $T' \in \mathsf{MST}(G, w_i)$ is the MST that has maximum overlap with H, we have $T' \in \mathsf{MST}(G, w)$ (because the only difference between w and w_i is that the weight of every edge in H is decreased by ϵ). Therefore, both T and T' are MSTs under w. Recall that both T and T' are MSTs under w_i, which gives $w_i(T) = w_i(T')$. Then we have $w(T) = w_i(T) - t \cdot \epsilon$ while $w(T') \leq w_i(T') - (t+1) \cdot \epsilon$, which is a contradiction. ■

5.3 Submodularity of the Progress Function

In this subsection, we prove Lemma 3 by showing the submodularity of the progress function. Consider any agent

$i \in K$. Before proving Lemma 3, we first show some properties of function f_i. For any subset $H \subseteq E$, let

$$\mathsf{MST}_i^H := \{\, T \in \mathsf{MST}_i : f_i(H) = |T \cap H| \,\}$$

be the set of MSTs that contain exactly $f_i(H)$ edges in H. We observe the following simple facts.

Observation 1. *For all edge $e \in E \setminus H$, we have $f_i(H \cup \{e\}) - f_i(H) \in \{0, 1\}$.*

Observation 2. *For all edge $e \in E \setminus H$, $f_i(H \cup \{e\}) - f_i(H) = 1$ if and only if edge $e \in T$ for some spanning tree $T \in \mathsf{MST}_i^H$.*

Observation 3. *For all edge $e \in H$, $f_i(H \setminus \{e\}) = f_i(H) - 1$ if and only if $e \in T$ for all $T \in \mathsf{MST}_i^H$.*

Proof of Lemma 3: Consider any agent $i \in K$. We prove that f_i is submodular by contradiction. Suppose that f_i is not submodular. By definition of submodularity, there must exist a subset $S \subsetneq E$ and two edges $a, b \in E \setminus S$ such that

$$f_i(S \cup \{a\}) - f_i(S) < f_i(S \cup \{a, b\}) - f_i(S \cup \{b\}). \quad (1)$$

By Observation 1, the above inequality implies $f_i(S \cup \{a\}) = f_i(S)$ and $f_i(S \cup \{a, b\}) = f_i(S \cup \{b\}) + 1$. Reordering Eq. (1) gives

$$f_i(S \cup \{b\}) - f_i(S) < f_i(S \cup \{a, b\}) - f_i(S \cup \{a\}), \quad (2)$$

which implies that $f_i(S \cup \{b\}) = f_i(S)$ and $f_i(S \cup \{a, b\}) = f_i(S \cup \{a\}) + 1$. Thus, we have

$$f_i(S) = f_i(S \cup \{a\}) = f_i(S \cup \{b\}) = f_i(S \cup \{a, b\}) - 1. \quad (3)$$

By Observation 2, the first equality of Eq. (3) implies that $a \notin T$ for all $T \in \mathsf{MST}_i^S$. By Observation 3, the last two equations of Eq. (3) implies that for all $T \in \mathsf{MST}_i^{S \cup \{a,b\}}$, we have $\{a, b\} \subseteq T$. Fix any $T \in \mathsf{MST}_i^{S \cup \{a,b\}}$, and consider the graph $G' = (V, T \setminus \{a, b\})$. Note that G' has three connected components, say with nodes C, C_a and C_b, where $a \in \mathcal{C}(C_a, C \cup C_b)$ and $b \in \mathcal{C}(C_a \cup C, C_b)$ (see Fig. 3). Here we use $\mathcal{C}(X, V \setminus X) = E \cap (X \times (V \setminus X))$ to denote the set of cut edges between X and $V \setminus X$. Observe that

(i) there does not exist $e \in \mathcal{C}(C_a, C \cup C_b)$ such that $e \prec_i a$; otherwise, T is strictly worse than $T \cup \{e\} \setminus \{a\}$ and is not an MST under $\preceq_i$;

(ii) for all $e \in \mathcal{C}(C_a, C \cup C_b) \cap (S \cup \{b\})$, we must have $a \prec_i e$; otherwise, we obtain another MST $T^* = T \cup \{e\} \setminus \{a\}$ such that $T^* \in \mathsf{MST}_i^{S \cup \{a,b\}}$, which contradicts the fact that $\{a, b\} \subseteq T$ for all $T \in \mathsf{MST}_i^{S \cup \{a,b\}}$.

Fix any $T' \in \mathsf{MST}_i^S$ and consider the graph $G'' = (V, T' \cup \{a\})$. Recall that $a \notin T'$ and thus G'' contains a cycle. Let e be any edge other than a from cut $\mathcal{C}(C_a, C \cup C_b)$ that also appears in the cycle (see Fig. 3). Note that we cannot have $a \prec_i e$ because otherwise the MST T' can be improved (by removing e and including a), which is a contradiction. By statement (i) from above, we cannot have $e \prec_i a$, and thus $a \sim_i e$; by statement (ii) from above, we have $e \notin S$. Therefore, $T^* = T' \cup \{a\} \setminus \{e\}$ is another MST, and contains the same number of edges from S as T', i.e., $T^* \in \mathsf{MST}_i^S$. However, this is again a contradiction because we have proved that for all $T \in \mathsf{MST}_i^S$, $a \notin T$. ■

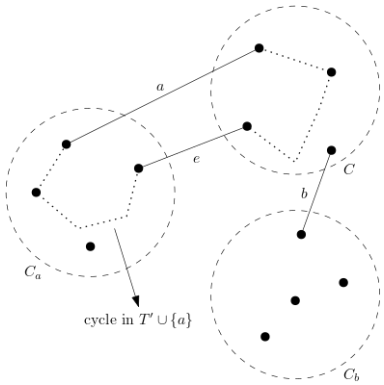

Figure 3: An illustration for the proof of Lemma 3. After removing edge a and b from T, we have three connected components C_a, C and C_b. Now, we consider an MST $T' \in \mathsf{MST}_i^S$. After adding edge a to T', there will be a cycle in $T' \cup \{a\}$ and thus there exists an edge e other than a in both the cycle and the cut $\mathcal{C}(C_a, C \cup C_b)$. By the property of the minimum spanning tree, we prove that $e \sim_i a$.

6 Conclusion and Open Problems

In this paper, we propose the multiagent MST cover problem in which we need to compute a minimum subgraph of an input graph that satisfies a group of k heterogeneous agents, where each agent is satisfied if the subgraph contains an MST under its own weight function. When the optimal solution of the problem is a spanning tree, the optimal solution can be computed efficiently. However, in the general case, no algorithm can do better than $((1-o(1)) \ln k)$-approximation unless $\mathsf{P} = \mathsf{NP}$. We then propose an $\mathcal{O}(\ln k)$-approximation algorithm, achieving the best possible approximation ratio under the assumption that $\mathsf{P} \neq \mathsf{NP}$. Further, our algorithm can be naturally generalized to the case when the MST constraints are generalized to maximum Matroid rank constraints, and the cardinality objective function is generalized to submodular.

Our work leaves several interesting problems open. For example, one possible direction is to relax the satisfaction constraint of the agents. In our work, we require that every agent is fully satisfied, e.g., can find an MST in the solution. It would be interesting to investigate whether the approximation ratio can be improved if we allow the solution to contain only approximate MST for each agent. However, note that the approximation must be additive, as we can change the reduction from the proof of Theorem 1 to show that the inapproximability remains the same even if the value of MST is 0 for all agents in K. Additionally, it would be interesting to study the case when agents' constraints are defined by combinatorial structures other than MST.

Acknowledgements

Chenyang Xu is funded by Science and Technology Innovation 2030 – "The Next Generation of Artificial Intelligence" Major Project No.2018AAA0100900. Xiaowei Wu is funded by FDCT (File no. 0143/2020/A3, SKL-IOTSC-2021-2023), the SRG of University of Macau (File no.

SRG2020-00020-IOTSC) and GDST (2020B1212030003). Bo Li is funded by HKSAR RGC (No. PolyU 25211321), NSFC (No. 62102333), CCF-HuaweiLK2022006, and PolyU Start-up (No. P0034420). The authors thank the anonymous reviewers for their comments.

References

Ahuja, R. K.; Magnanti, T. L.; and Orlin, J. B. 1993. *Network flows - theory, algorithms and applications*. Prentice Hall.

Amorosi, L.; and Puerto, J. 2022. Two-phase strategies for the bi-objective minimum spanning tree problem. *Int. Trans. Oper. Res.*, 29(6): 3435–3463.

Anshelevich, E.; Bhardwaj, O.; Elkind, E.; Postl, J.; and Skowron, P. 2018. Approximating optimal social choice under metric preferences. *Artif. Intell.*, 264: 27–51.

Anshelevich, E.; Filos-Ratsikas, A.; Shah, N.; and Voudouris, A. A. 2021. Distortion in Social Choice Problems: The First 15 Years and Beyond. In *IJCAI*, 4294–4301. ijcai.org.

Bachhiesl, P.; Paulus, G.; Prossegger, M.; Werner, J.; and Stögner, H. 2002. Cost Optimized Layout of Fibre Optic Networks in the Access Net Domain. In *OR*, 247–252. Springer.

Caragiannis, I.; Nath, S.; Procaccia, A. D.; and Shah, N. 2017. Subset Selection Via Implicit Utilitarian Voting. *J. Artif. Intell. Res.*, 58: 123–152.

Caragiannis, I.; and Procaccia, A. D. 2011. Voting almost maximizes social welfare despite limited communication. *Artif. Intell.*, 175(9-10): 1655–1671.

Conitzer, V.; Freeman, R.; and Shah, N. 2017. Fair Public Decision Making. In *EC*, 629–646. ACM.

Correia, P.; Paquete, L.; and Figueira, J. R. 2021. Finding multi-objective supported efficient spanning trees. *Comput. Optim. Appl.*, 78(2): 491–528.

Darmann, A. 2016. It is difficult to tell if there is a Condorcet spanning tree. *Math. Methods Oper. Res.*, 84(1): 93–104.

Darmann, A.; Klamler, C.; and Pferschy, U. 2009. Maximizing the minimum voter satisfaction on spanning trees. *Math. Soc. Sci.*, 58(2): 238–250.

de Sousa, E. G.; Santos, A. C.; and Aloise, D. J. 2015. An exact method for solving the bi-objective Minimum Diameter-Cost Spanning Tree Problem. *RAIRO Oper. Res.*, 49(1): 143–160.

Ehrgott, M. 2005. *Multicriteria Optimization (2. ed.)*. Springer.

Escoffier, B.; Gourvès, L.; and Monnot, J. 2013. Fair solutions for some multiagent optimization problems. *Auton. Agents Multi Agent Syst.*, 26(2): 184–201.

Fain, B.; Munagala, K.; and Shah, N. 2018. Fair Allocation of Indivisible Public Goods. In *EC*, 575–592. ACM.

Fernandes, I. F. C.; Maia, S. M. D. M.; Goldbarg, E. F. G.; and Goldbarg, M. C. 2019. A Multi-agent Transgenetic Algorithm for the Bi-objective Spanning Tree Problem. In *LAGOS*, volume 346 of *Electronic Notes in Theoretical Computer Science*, 449–460. Elsevier.

Fluschnik, T.; Skowron, P.; Triphaus, M.; and Wilker, K. 2019. Fair Knapsack. In *AAAI*, 1941–1948. AAAI Press.

Garg, J.; Kulkarni, P.; and Murhekar, A. 2021. On Fair and Efficient Allocations of Indivisible Public Goods. In *FSTTCS*, volume 213 of *LIPIcs*, 22:1–22:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Gkatzelis, V.; Halpern, D.; and Shah, N. 2020. Resolving the Optimal Metric Distortion Conjecture. In *FOCS*, 1427–1438. IEEE.

Gourvès, L.; Monnot, J.; and Tlilane, L. 2015. Approximate tradeoffs on weighted labeled matroids. *Discret. Appl. Math.*, 184: 154–166.

Hamacher, H. W.; and Ruhe, G. 1994. On spanning tree problems with multiple objectives. *Ann. Oper. Res.*, 52(4): 209–230.

Hassin, R.; and Levin, A. 2004. An efficient polynomial time approximation scheme for the constrained minimum spanning tree problem using matroid intersection. *SIAM J. Comput.*, 33(2): 261–268.

Hong, S.; Chung, S.; and Park, B. H. 2004. A fully polynomial bicriteria approximation scheme for the constrained spanning tree problem. *Oper. Res. Lett.*, 32(3): 233–239.

Kruskal, J. B. 1956. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1): 48–50.

Lai, X.; and Xia, X. 2019. Multi-objective evolutionary algorithm on simplified bi-objective minimum weight minimum label spanning tree problems. *Int. J. Comput. Sci. Eng.*, 20(3): 354–361.

Li, B.; Wu, X.; Xu, C.; and Zhang, R. 2022. Multiagent MST Cover: Pleasing All Optimally via A Simple Voting Rule. *CoRR*, abs/2211.13578.

Majumder, S.; Barma, P. S.; Biswas, A.; Banerjee, P.; Mandal, B. K.; Kar, S.; and Ziemba, P. 2022. On Multi-Objective Minimum Spanning Tree Problem under Uncertain Paradigm. *Symmetry*, 14(1): 106.

Munagala, K.; and Wang, K. 2019. Improved Metric Distortion for Deterministic Social Choice Rules. In *EC*, 245–262. ACM.

Ohlsson, H.; Gustafsson, O.; and Wanhammar, L. 2004. Implementation of low complexity FIR filters using a minimum spanning tree. In *Proceedings of the 12th IEEE Mediterranean Electrotechnical Conference*, 261–264. IEEE.

Papadimitriou, C. H.; and Yannakakis, M. 2000. On the Approximability of Trade-offs and Optimal Access of Web Sources. In *FOCS*, 86–92. IEEE Computer Society.

Prakash, V. P.; Patvardhan, C.; and Srivastav, A. 2020. A novel Hybrid Multi-objective Evolutionary Algorithm for the bi-Objective Minimum Diameter-Cost Spanning Tree (bi-MDCST) problem. *Eng. Appl. Artif. Intell.*, 87.

Procaccia, A. D.; and Rosenschein, J. S. 2006. The Distortion of Cardinal Preferences in Voting. In *CIA*, volume 4149 of *Lecture Notes in Computer Science*, 317–331. Springer.

Ruzika, S.; and Hamacher, H. W. 2009. A Survey on Multiple Objective Minimum Spanning Tree Problems. In *Algorithmics of Large and Complex Networks*, volume 5515 of *Lecture Notes in Computer Science*, 104–116. Springer.

Wolsey, L. A. 1982. An analysis of the greedy algorithm for the submodular set covering problem. *Comb.*, 2(4): 385–393.