An ETH-Tight Algorithm for Multi-Team Formation

Daniel Lokshtanov ✉
University of California, Santa Barbara, CA, USA

Saket Saurabh ✉
The Institute of Mathematical Sciences (HBNI), Chennai, India
University of Bergen, Norway

Subhash Suri ✉
University of California, Santa Barbara, CA, USA

Jie Xue ✉
New York University Shanghai, China

Abstract

In the Multi-Team Formation problem, we are given a ground set C of n candidates, each of which is characterized by a d-dimensional attribute vector in $\mathbb{R}^d$, and two positive integers α and β satisfying $\alpha\beta \leq n$. The goal is to form α disjoint teams $T_1, ..., T_\alpha \subseteq C$, each of which consists of β candidates in C, such that the total score of the teams is maximized, where the score of a team T is the sum of the h_j maximum values of the j-th attributes of the candidates in T, for all $j \in \{1, ..., d\}$. Our main result is an $2^{2^{O(d)}} n^{O(1)}$-time algorithm for Multi-Team Formation. This bound is ETH-tight since a $2^{2^{d/c}} n^{O(1)}$-time algorithm for any constant $c > 12$ can be shown to violate the Exponential Time Hypothesis (ETH). Our algorithm runs in polynomial time for all dimensions up to $d = c \log \log n$ for a sufficiently small constant $c > 0$. Prior to our work, the existence of a polynomial time algorithm was an open problem even for $d = 3$.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Team formation, Parameterized algorithms, Exponential Time Hypothesis

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2021.28

Acknowledgements We would like to thank an anonymous reviewer for pointing us to the statement of [10], allowing us to drastically simplify a previous version of the paper.

1 Introduction

The problem of team formation arises in many organizational settings – project management, product development, team sports, academic committees, legal defence teams, to name a few – and remains an important area of research in mathematical social sciences [12, 16, 22, 24]. Within computer science and operations research, several application domains – distributed robotics, AI, multi-agent systems, online crowdsourcing, databases – also use team formation models for execution of complex tasks that require cooperation or coalition of multiple agents with different capabilities [5, 18, 21, 23]. The basic setting of a Team Formation problem includes a ground set C of n candidates and a number $\beta \leq n$. The goal is to form a team $T \subseteq C$ of a size β such that $\mathsf{scr}(T)$ is maximized, where $\mathsf{scr}(\cdot)$ is a pre-defined scoring function. A concrete example of a scoring function frequently used in the literature [11, 25] (often in conjunction with other, more complex measures of team performance) is the *skill coverage* function. There is a set U of useful skills, each candidate $a \in C$ has a subset S_a of these skills, and we evaluate the team by the number of different skills covered by the team members. In other words, $\mathsf{scr}(T) = |\bigcup_{a \in T} S_a|$. It is easy to see that Team Formation with the skill coverage scoring function is equivalent to the-well studied Maximum Coverage problem, which is NP-complete [7], admits a $(1 - \frac{1}{e})$-approximation algorithm [15], and is NP-hard to approximate [4] within any factor smaller than $1 - \frac{1}{e}$.

A natural generalization of Team Formation is the Multi-team Formation problem, where we want to form α disjoint teams $T_1, \ldots, T_\alpha \subseteq C$ each of size β that collectively maximize the total score $\sum_{i=1}^{\alpha} \mathsf{scr}(T_i)$. This generalization is well-motivated in practice: in many applications, we want to form multiple teams from a common pool of candidates, where candidate can belong to at most one team. Multi-team Formation has some resemblance to the *coalition structure generation* problem in multi-agent systems and AI, where the goal is to partition a set of candidates into groups, called coalitions [19]. However, in these applications, the scoring function for evaluating a coalition is assumed to be an arbitrary black box function. As a result, the size of each team (coalition) is not explicitly specified but rather determined by the objective function of maximizing the total coalition structure value – e.g. if putting all the candidates into a single coalition maximizes the total value, then that is the optimal solution. In [14], a dynamic programming algorithm is described for computing an optimal coalition structure in time $O(3^n)$. Unlike the (single) Team Formation problem, Multi-team Formation has not yet received much attention, and beyond the exponential bound of Michalak et al. [14], no algorithmic result appears to be known for forming multiple teams except for the recent work of Schibler et al. [20].

In this paper, we follow Schibler et al. [20] and investigate Multi-team Formation with a fundamental scoring function, called *sum-of-maxima* scoring, to be defined below. A common model for characterizing a candidate is a multi-dimensional attribute vector in which each entry measures a certain performance of the candidate. For instance, in college admissions, such a vector may include scores of different standardized tests, grade point averages, etc. In project management, the categories may include various technical skills as well as non-technical attributes such as leadership qualities. Following Page's influential work on team performance [16], it is generally acknowledged that simply adding up all the scores is a poor measure of team performance – instead, strength in multiple dimensions (skill diversity) is essential. When the candidates are characterized by attribute vectors, one natural scoring is to take the *best attribute* of the candidates in the team T in each dimension and set the score of T to be the sum of these best attributes. Kleinberg and Raghu [8], in their work on team performance metrics and testing, suggested extending this further to sum-of-top-h scores in each dimension, for some $h \leq \beta$, ensuring both coverage of all the skills (dimensions) and robustness (no single point of failure). We allow a slightly more general scoring rule, where for each dimension j, a possibly different number h_j of top attributes are considered. We call this the *sum-of-maxima* scoring. Formally, each candidate $a \in C$ is characterized by a d-dimensional attribute vector $(\kappa_1(a), \ldots, \kappa_d(a)) \in \mathbb{R}^d$. For a given vector $\mathbf{h} = (h_1, \ldots, h_d) \in \mathbb{Z}_+^d$, the sum-of-$\mathbf{h}$-maxima scoring function is defined as

$$\mathsf{som}_{\mathbf{h}}(T) = \sum_{j=1}^{d} \max^{h_j} \{\kappa_j(a) : a \in T\}, \tag{1}$$

where the notation $\max^{h_j} S$ denotes the sum of the largest h_j numbers in the *multiset* S of numbers (if $|S| < h_j$, then $\max^{h_j} S$ is the sum of all numbers in S). It is easy to see that the sum-of-maxima scoring function generalizes skill coverage. In particular, the Maximum Coverage problem is a special case of Multi-team Formation where $\mathbf{h}$ is the vector of all 1's, all the candidate attributes are binary, and $\alpha = 1$.

In the rest of the paper, the Multi-team Formation problem we discuss is always with respect to sum-of-maxima scoring. Since Multi-team Formation generalizes Maximum Coverage, it is clearly NP-hard (when the dimension d is unbounded). Schibler et al. [20]

proved that Multi-team Formation is NP-hard when $d = \Theta(\log n)$, even with binary attributes and team size $\beta \geq 4$. These hardness claims, however, depend on the rather unrealistic assumption that the dimension d of attribute vectors must be quite large – in most applications, the number of attributes (e.g., standardized test scores) is much more modest. Therefore, it is interesting to study the complexity of Multi-team Formation when d is small. Indeed, Schibler et al. [20] gave a polynomial-time algorithm for the case of $d = 2$ and leave as an open problem whether the problem is polynomial time solvable for any constant $d \geq 3$. Our main result is a new algorithm for Multi-team Formation, which runs in polynomial time for any $d \leq c \cdot \log\log n$ where $c > 0$ is a sufficiently small constant (and hence for any constant d). Specifically, we prove the following theorem.

▶ **Theorem 1.** *There exists a $2^{2^{O(d)}} n^{O(1)}$-time algorithm for Multi-team Formation.*

In the view of Parameterized Complexity, this is the first Fixed-Parameter Tractable (FPT) algorithm for Multi-team Formation parameterized by the dimension d. The analysis of the algorithm of Theorem 1 involves a novel application of *Graver Bases*, a notion that has successfully been applied to yield fixed parameter tractability results for a number of problems in Mathematical Programming. To the best of our understanding, however, none of the existing state-of-the art results [3, 6, 9] can be applied in a black box fashion to yield an FPT algorithm for Multi-team Formation parameterized by d. It remains an interesting research question to generalize Theorem 1 to an FPT algorithm for solving a class of mathematical programs that is powerful enough to encompass Multi-team Formation.

The time complexity of our algorithm grows double exponentially with d and, under plausible complexity theoretic assumptions, it cannot be substantially improved. In particular, a fresh look at the NP-hardness reduction of Schibler et al. [20] reveals that any algorithm that solves Multi-team Formation in $2^{2^{d/c}} \cdot n^{O(1)}$ time for a sufficiently large constant c will violate the Exponential Time Hypothesis (ETH).

▶ **Theorem 2.** *The existence of a $2^{2^{d/c}} n^{O(1)}$-time algorithm for Multi-team Formation with any constant $c > 12$ violates the Exponential Time Hypothesis (ETH).*

Therefore, our algorithm is ETH-tight, and adds Multi-team Formation to the small club of problems (together with Edge Clique Cover [2] and Distinct Vectors [17]) for which both a double exponential time algorithm and a double exponential time lower bound were known.

2 An ETH-tight algorithm

In this section, we present our algorithm for Multi-team Formation in Theorem 1, and also prove Theorem 2 (which is easy). We begin by introducing some basic notations. Let $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Z}_+$, $\mathbb{R}$ to denote the set of natural numbers (including 0), integers, positive integers, and real numbers, respectively. For two vectors $\mathbf{u}, \mathbf{v}$ of the same dimension, we use $\langle \mathbf{u}, \mathbf{v} \rangle$ to denote the inner product of $\mathbf{u}, \mathbf{v}$. For a number $k \in \{0, \dots, 2^d - 1\}$, let $\mathsf{bin}(k)$ be the d-bit binary representation of k, which is a d-dimensional binary vector, and $\mathsf{bin}_j(k)$ be the j-th entry of $\mathsf{bin}(k)$, i.e., the j-th (highest) digit of the d-bit binary representation of k.

Recall that in Multi-team Formation, the input includes a set C of n candidates where each $a \in C$ is characterized by a d-dimensional attribute vector $\kappa(a) = (\kappa_1(a), \dots, \kappa_d(a)) \in \mathbb{R}^d$, a vector $\mathbf{h} = (h_1, \dots, h_d) \in \mathbb{Z}_+^d$ used for defining the scoring function $\mathsf{som}_{\mathbf{h}}$, and two integers $\alpha, \beta > 0$ satisfying $\alpha\beta \leq n$. The goal is to form α disjoint teams $T_1, \dots, T_\alpha \subseteq C$ of size β such that $\sum_{i=1}^{\alpha} \mathsf{som}_{\mathbf{h}}(T_i)$ is maximized. Without loss of generality, we may assume that $h_j \leq \beta$ for all $j \in \{1, \ldots, d\}$, because $\mathsf{som}_{\mathbf{h}}(T)$ remains unchanged for all $T \subseteq C$ with $|T| = \beta$ if we replace all $h_j > \beta$ with β, as one can easily verified. Let opt denote the optimum of the input instance.

Consider a solution $T_1, \ldots, T_\alpha \subseteq C$ of the problem. The total score of this solution, $\sum_{i=1}^{\alpha} \mathsf{som}_{\mathbf{h}}(T_i)$, is the sum of some attributes $\kappa_j(a)$ for $a \in \bigcup_{i=1}^{\alpha} T_i$. For each team T_i, each candidate $a \in T_i$ contributes to the score $\mathsf{som}_{\mathbf{h}}(T_i)$ in a certain way. Specifically, for each dimension $j \in \{1, \ldots, d\}$, the candidate a is either among the top h_j candidates in T_i in that dimension, in which case it contributes $\kappa_j(a)$, or it is not, in which case it contributes 0[1]. The information of how the d attributes of $a \in T_i$ contribute to the score $\mathsf{som}_{\mathbf{h}}(T_i)$ can be depicted by a number $k \in \{0, \ldots, 2^d - 1\}$ (or equivalently, a d-bit binary string) where $\mathsf{bin}_j(k) = 1$ if $\kappa_j(a)$ contributes to $\mathsf{som}_{\mathbf{h}}(T_i)$ and $\mathsf{bin}_j(k) = 0$ if $\kappa_j(a)$ does not contribute, for $j \in \{1, \ldots, d\}$. We call k the *type* of the candidate a in the solution $T_1, \ldots, T_\alpha$. Now every candidate in $\bigcup_{i=1}^{\alpha} T_i$ has its type, which is a number in $\{0, \ldots, 2^d - 1\}$. For the unassigned candidates, i.e., the candidates in $C \setminus \bigcup_{i=1}^{\alpha} T_i$, we simply say their type is $\Box$ (in the solution $T_1, \ldots, T_\alpha$). In this way, we give every candidate in C a type in the solution, which is an element in $\mathit{\Gamma} = \{0, \ldots, 2^d - 1\} \cup \{\Box\}$. We then define the *type assignment* (or *assignment* for short) of the solution $T_1, \ldots, T_\alpha$ as the function $\pi : C \to \mathit{\Gamma}$ that maps each candidate to its type in the solution.

We consider the following question: for a solution $T_1, \ldots, T_\alpha \subseteq C$, if we were only given its type assignment $\pi : C \to \mathit{\Gamma}$ without the original teams $T_1, \ldots, T_\alpha$, how much information about $T_1, \ldots, T_\alpha$ can we recover from π? Observe first that we *can* easily recover the total score $\sum_{i=1}^{\alpha} \mathsf{som}_{\mathbf{h}}(T_i)$ of the solution, simply because the types of the candidates record how their attributes contribute to the total score. Specifically, if we define

$$\mathsf{scr}(\pi) = \sum_{a \in C,\ \pi(a) \neq \Box} \langle \mathsf{bin}(\pi(a)), \kappa(a) \rangle = \sum_{a \in C,\ \pi(a) \neq \Box} \left(\sum_{j=1}^{d} \mathsf{bin}_j(\pi(a)) \cdot \kappa_j(a) \right), \qquad (2)$$

which we call the *score* of π, then it is clear that $\sum_{i=1}^{\alpha} \mathsf{som}_{\mathbf{h}}(T_i) = \mathsf{scr}(\pi)$. At the same time, however, we *cannot* recover the teams $T_1, \ldots, T_\alpha$ from π, because it can happen that different solutions share the same type assignment (for example, there are situations where two candidates with the same attributes, but in different teams, could be swapped without changing their type, leading to a different solution with the same type assignment).

We say a solution $T_1, \ldots, T_\alpha \subseteq C$ *realizes* a type assignment function $\pi : C \to \mathit{\Gamma}$ if π is the type assignment of $T_1, \ldots, T_\alpha$. Thus, for a type assignment function $\pi : C \to \mathit{\Gamma}$, there could be zero, one, or more solutions that realize it, and all such solutions have the same total score. We say π is *realizable* if there exists at least one solution that realizes π. What we want is essentially a realizable $\pi : C \to \mathit{\Gamma}$ that maximizes $\mathsf{scr}(\pi)$.

Note that there are too many (type assignment) functions $\pi : C \to \mathit{\Gamma}$ to go over all of them; indeed, the number of such functions is $(2^d + 1)^n$. Furthermore, it turns out to be difficult to check whether a given π is realizable, and even if we know π is realizable, it is not clear how to find a witness solution $T_1, \ldots, T_\alpha \subseteq C$ that realizes π. For this reason, our algorithm does not work on type assignment functions directly. Instead, we only guess some distinguishing features of the type assignment of an optimal solution. Perhaps the most natural distinguishing feature is "how many candidates are there of each type". We

1 Here we assume that the "sum-of-top-h_j" function $\max^{h_j}$ in Equation 1 breaks ties in a certain way (e.g., take the attributes of the candidates with smaller indices first, etc.) so that the contributing attributes of each candidate in the team is uniquely defined.

formalize this as follows. The *configuration* of a function $\pi : C \to \Gamma$ is a 2^d-dimensional vector $\mathsf{conf}(\pi) = (c_0, \dots, c_{2^d-1}) \in \mathbb{N}^{2^d}$ where $c_k = |\pi^{-1}(\{k\})|$ for $k \in \{0, \dots, 2^d - 1\}$. In other words, the k-th entry c_k of the vector $\mathsf{conf}(\pi)$ records the number of candidates assigned to type k by π.

Clearly, not every vector in $\mathbb{N}^{2^d}$ can be the configuration of some realizable function. Next, we establish a simple *necessary* (but not sufficient) condition for a vector to be the configuration of some realizable function. Suppose $\mathbf{c} = (c_0, \dots, c_{2^d-1})$ is the configuration of a realization function $\pi : C \to \Gamma$ and let $T_1, \dots, T_\alpha \subseteq C$ be the solution that realizes π, i.e., π is the type assignment of $T_1, \dots, T_\alpha$. For $i \in \{1, \dots, \alpha\}$ and $k \in \{0, \dots, 2^d - 1\}$, let $v_{i,k}$ be the number of candidates in T_i which are mapped to k by π, i.e., $v_{i,k} = |\pi^{-1}(\{k\}) \cap T_i|$. Since π maps all candidates in $C \backslash (\bigcup_{i=1}^{\alpha} T_i)$ to $\square$, we have $c_k = |\pi^{-1}(\{k\})| = \sum_{i=1}^{\alpha} v_{i,k}$ for all $k \in \{0, \dots, 2^d - 1\}$ and hence $\mathbf{c} = \sum_{i=1}^{\alpha} \mathbf{v}_i$ where $\mathbf{v}_i = (v_{i,0}, \dots, v_{i,2^d-1})$. Now what are the conditions that each $\mathbf{v}_i$ has to satisfy? First, since $|T_i| = \beta$ and π maps all candidates in T_i to $\{0, \dots, 2^d - 1\}$, the sum of all entries of $\mathbf{v}_i$ is equal to β, i.e., $\sum_{k=0}^{2^d-1} v_{i,k} = \beta$. Second, for each $j \in \{1, \dots, d\}$, the number of candidates in T_i which contribute in the j-th dimension is precisely h_j, and thus the sum of the entries of $\mathbf{v}_i$ corresponding to types k which contribute in the j-th dimension, i.e., $\mathsf{bin}_j(k) = 1$, is equal to h_j, i.e., $\sum_{k=0}^{2^d-1} v_{i,k} \cdot \mathsf{bin}_j(k) = h_j$. To summarize, in order to be the configuration of some realizable function, a vector $\mathbf{c}$ must be the sum of α vectors each of which satisfies the above two conditions. This is exactly the necessary condition we want. Formally, we give the following definition.

▶ **Definition 3** (legal vectors). *A vector $\mathbf{v} = (v_0, \dots, v_{2^d-1}) \in \mathbb{N}^{2^d}$ is* $(\boldsymbol{\beta}, \mathbf{h})$***-legal*** *(or simply* ***legal*** *when β and $\mathbf{h}$ are all clear from the context) if $\sum_{k=0}^{2^d-1} v_k = \beta$ and $\sum_{k=0}^{2^d-1} v_k \cdot \mathsf{bin}_j(k) = h_j$ for all $j \in \{1, \dots, d\}$.*

▶ **Fact 4.** *If $\pi : C \to \Gamma$ is realizable, then $\mathsf{conf}(\pi)$ is the sum of α legal vectors.*

Note that the converse of the above fact is not true, i.e., it is possible that $\mathsf{conf}(\pi)$ is the sum of α legal vectors but π is not the type assignment of any solution. However, we have the following nice property.

▶ **Lemma 5.** *If $\pi : C \to \Gamma$ is a function such that $\mathsf{conf}(\pi)$ is the sum of α legal vectors, then $\mathsf{scr}(\pi) \leq \mathsf{opt}$. Furthermore, given π and a decomposition $\mathsf{conf}(\pi) = \sum_{i=1}^{\alpha} \mathbf{v}_i$ into legal vectors, one can compute in $O(n + 2^d)$ time a solution $T_1, \dots, T_\alpha \subseteq C$ of the problem such that $\mathsf{scr}(\pi) \leq \sum_{i=1}^{\alpha} \mathsf{som}_{\mathbf{h}}(T_i)$.*

Proof. Suppose $\mathsf{conf}(\pi) = (c_0, \dots, c_{2^d-1}) = \sum_{i=1}^{\alpha} \mathbf{v}_i$, where each $\mathbf{v}_i = (v_{i,0}, \dots, v_{i,2^d-1})$ is a legal vector. For $k \in \{0, \dots, 2^d - 1\}$, we arbitrarily partition the c_k candidates in $\pi^{-1}(\{k\})$ into α groups $G_{1,k}, \dots, G_{\alpha,k}$ such that $|G_{i,k}| = v_{i,k}$; this is possible because $c_k = \sum_{i=1}^{\alpha} v_{i,k}$. We then define $T_i = \bigcup_{k=0}^{2^d-1} G_{i,k}$ for $i \in \{1, \dots, \alpha\}$. It is clear that $T_1, \dots, T_\alpha$ are disjoint subsets of C with size β. Therefore, $\sum_{i=1}^{\alpha} \mathsf{som}_{\mathbf{h}}(T_i) \leq \mathsf{opt}$. It suffices to show $\mathsf{scr}(\pi) \leq \sum_{i=1}^{\alpha} \mathsf{som}_{\mathbf{h}}(T_i)$. Note that $\pi(a) \in \{0, \dots, 2^d - 1\}$ for all $a \in \bigcup_{i=1}^{\alpha} T_i$ and $\pi(a) = \square$ for all $a \in C \backslash (\bigcup_{i=1}^{\alpha} T_i)$. So we have $\mathsf{scr}(\pi) = \sum_{i=1}^{\alpha} \sum_{a \in T_i} \sum_{j=1}^{d} \mathsf{bin}_j(\pi(a)) \cdot \kappa_j(a)$. Equivalently, $\mathsf{scr}(\pi) = \sum_{i=1}^{\alpha} \sum_{j=1}^{d} \sum_{a \in T_{i,j}} \kappa_j(a)$, where $T_{i,j} = \{a \in T_i : \mathsf{bin}_j(\pi(a)) = 1\}$. Since $\mathbf{v}_1, \dots, \mathbf{v}_\alpha$ are $(\beta, \mathbf{h})$-legal, we have $|T_{i,j}| = h_j$ for all $i \in \{1, \dots, \alpha\}$ and $j \in \{1, \dots, d\}$. Thus, $\sum_{a \in T_{i,j}} \kappa_j(a) \leq \max^{h_j} \{\kappa_j(a) : a \in T_i\}$ (recall that $\max^{h_j} S$ denotes the sum of the largest h_j numbers in the multiset S). It follows that

$$\mathsf{scr}(\pi) = \sum_{i=1}^{\alpha} \sum_{j=1}^{d} \sum_{a \in T_{i,j}} \kappa_j(a) \leq \sum_{i=1}^{\alpha} \sum_{j=1}^{d} \max^{h_j} \{\kappa_j(a) : a \in T_i\} = \sum_{i=1}^{\alpha} \mathsf{som}_{\mathbf{h}}(T_i).$$

Therefore, $\mathsf{scr}(\pi) \leq \mathsf{opt}$. If we are given π and the legal vectors $\mathbf{v}_1, \dots, \mathbf{v}_\alpha$, then the teams $T_1, \dots, T_\alpha$ can clearly be constructed in $O(n + 2^d)$ time. ◀

With the above lemma in hand, it now suffices to compute a function $\pi^* : C \to \mathit{\Gamma}$ with the maximum $\mathsf{scr}(\pi^*)$ such that $\mathsf{conf}(\pi^*)$ is the sum of α legal vectors and a decomposition $\mathsf{conf}(\pi^*) = \sum_{i=1}^{\alpha} \mathbf{v}_i$ into legal vectors. Indeed, once we have the function π^* and the decomposition $\mathsf{conf}(\pi^*) = \sum_{i=1}^{\alpha} \mathbf{v}_i$, we can apply the above lemma to obtain a solution $T_1^*, \dots, T_\alpha^* \subseteq C$ satisfying $\mathsf{scr}(\pi^*) \leq \sum_{i=1}^{\alpha} \mathsf{som}_{\mathbf{h}}(T_i^*)$. Note that Fact 4 guarantees $\mathsf{scr}(\pi^*) \geq \mathsf{opt}$, which implies $\sum_{i=1}^{\alpha} \mathsf{som}_{\mathbf{h}}(T_i^*) \geq \mathsf{opt}$, i.e., $T_1^*, \dots, T_\alpha^*$ is an optimal solution.

Next, we show how to compute the function π^* and the decomposition efficiently. To this end, we formulate the problem as an integer linear programming (ILP) instance. For each candidate $a \in C$, we define $2^d + 1$ variables $u_0(a), \dots, u_{2^d-1}(a), u_\Box(a)$. These variables are used to encode the information of π^*. Specifically, the variable $u_k(a)$ will indicate whether $\pi^*(a) = k$: $u_k(a) = 1$ if $\pi^*(a) = k$ and $u_k(a) = 0$ if $\pi^*(a) \neq k$. Therefore, the values of these variables are in $\{0, 1\}$ and must satisfy the constraints $\sum_{k \in \mathit{\Gamma}} u_k(a) = 1$ for all $a \in C$. Our objective function, which is $\mathsf{scr}(\pi^*)$, can be expressed as $\sum_{a \in C} \sum_{k=0}^{2^d-1} u_k(a) \cdot \langle \mathsf{bin}(k), \kappa(a) \rangle$, according to the formula of Equation 2. In addition, we need variables and constraints to guarantee that $\mathsf{conf}(\pi^*)$ is the sum of α legal vectors. Note that $\mathsf{conf}(\pi^*)$ can be expressed as $\sum_{a \in C} \mathbf{u}(a)$, where $\mathbf{u}(a) = (u_0(a), \dots, u_{2^d-1}(a))$. We introduce variables $v_{i,0}, \dots, v_{i,2^d-1}$ for all $i \in \{1, \dots, \alpha\}$. Each vector $\mathbf{v}_i = (v_{i,0}, \dots, v_{i,2^d-1})$ is supposed to be a legal vector. So we include the constraints $\sum_{k=0}^{2^d-1} v_{i,k} = \beta$ and $\sum_{k=0}^{2^d-1} v_{i,k} \cdot \mathsf{bin}_j(k) = h_j$ for all $j \in \{1, \dots, d\}$. Finally, we need to constraint $\sum_{a \in C} \mathbf{u}(a) = \sum_{i=1}^{\alpha} \mathbf{v}_i$ to ensure that $\mathsf{conf}(\pi^*)$ is the sum of $\mathbf{v}_1, \dots, \mathbf{v}_\alpha$. In sum, our ILP instance is

$$\begin{aligned}
&\max \sum_{a \in C} \sum_{k=0}^{2^d-1} u_k(a) \cdot \langle \mathsf{bin}(k), \kappa(a) \rangle \\
\text{s.t.}\quad & \textstyle\sum_{k \in \mathit{\Gamma}} u_k(a) = 1 \text{ for all } a \in C, \\
& \textstyle\sum_{k=0}^{2^d-1} v_{i,k} = \beta \text{ for all } i \in \{1, \dots, \alpha\}, \\
& \textstyle\sum_{k=0}^{2^d-1} v_{i,k} \cdot \mathsf{bin}_j(k) = h_j \text{ for all } i \in \{1, \dots, \alpha\} \text{ and } j \in \{1, \dots, d\}, \\
& \textstyle\sum_{a \in C} \mathbf{u}(a) = \sum_{i=1}^{\alpha} \mathbf{v}_i, \\
& \mathbf{0} \leq \mathbf{u}(a) \leq \mathbf{1} \text{ for all } a \in C \text{ and } \mathbf{v}_i \geq \mathbf{0} \text{ for all } i \in \{1, \dots, \alpha\}.
\end{aligned} \tag{3}$$

The above ILP instance has $(2^d + 1)n + 2^d\alpha$ variables, thus we cannot apply any general ILP solver to solve it in time polynomial in n. Fortunately, this ILP instance has some nice structural property which we can exploit. In order to describe the property, we need to first introduce the notion of N-fold ILP. In an N-fold ILP instance, the linear constraints on the variable vector $\mathbf{x}$ can be represented as $\mathbf{x}_{\text{low}} \leq \mathbf{x} \leq \mathbf{x}_{\text{high}}$ and $A\mathbf{x} = \mathbf{b}$ where

$$A = \begin{pmatrix} M_1 & M_2 & \cdots & M_N \\ M_1' & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & M_2' & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & M_N' \end{pmatrix}. \tag{4}$$

Let r be the maximum number of rows of the matrices $M_1, \dots, M_N$ and $M_1', \dots, M_N'$, and t be the maximum number of columns of the matrices $M_1', \dots, M_N'$. It was shown in [10] that the N-fold ILP instance can be solved in $\Delta^{O(r^3)}(Nt)^{O(1)}$ time, where $\Delta = \max\{2, \|A\|_\infty\}$.

We observe that our ILP instance in Equation 3 is in fact an N-fold ILP instance with $N = n + \alpha$, $r = 2^d$, $t = 2^d + 1$, and $\Delta = 2$. To this end, we classify our variables into $n + \alpha$ groups. For each $a \in C$, we have a group $G_a = \{u_k(a) : k \in \mathit{\Gamma}\}$ of $2^d + 1$ variables. For each $i \in \{1,\dots,\alpha\}$, we have a group $G'_i = \{v_{i,0},\dots,v_{i,2^d-1}\}$ of 2^d variables. We obtain our variable vector $\mathbf{x}$ by permuting all $(2^d+1)n + 2^d\alpha$ variables such that the variables in each group are consecutive in the permutation. Now notice that the constraint $\sum_{k\in\Gamma} u_k(a) = 1$ is only for the variables in G_a, while the constraints $\sum_{k=0}^{2^d-1} v_{i,k} = \beta$ and $\sum_{k=0}^{2^d-1} v_{i,k}\cdot\mathsf{bin}_j(k) = h_j$ for $j \in \{1,\dots,d\}$ are only for the variables in G'_i. We call these constraints *local constraints*. Local constraints can be realized using the M'-matrices in Equation 4; the number of rows of these matrices is at most $d+1$ because we have one local constraint for each group G_a and $d+1$ local constraints for each group G'_i, and the number of columns of these matrices is at most 2^d+1 because each group has at most 2^d+1 variables. Finally, we have the "global" constraints $\sum_{a\in C} \mathbf{u}(a) = \sum_{i=1}^{\alpha} \mathbf{v}_i$. Since the dimension of the vectors $\mathbf{u}(a)$ and $\mathbf{v}_i$ is 2^d, the global constraints can be expressed as $M\mathbf{x} = \mathbf{0}$ for some 2^d-row matrix M, which can be in turn realized using matrices $M_1,\dots,M_N$ in Equation 4. To summarize, the constraints of our ILP instance of Equation 3 can be written as $A\mathbf{x} = \mathbf{b}$, where A is of the form of Equation 4 in which $N = n+\alpha$ and the maximum number of rows (resp., columns) of the matrices $M_1,\dots,M_N,M'_1,\dots,M'_N$ is 2^d (resp., 2^d+1). Also, as one can easily verified, the entries of A are all in $\{-1,0,1\}$, which implies $\|A\|_\infty \le 1$ and $\Delta = 2$. Therefore, applying the algorithm of [10] solves our ILP instance in $2^{2^{O(d)}} n^{O(1)}$ time.

After solving the ILP instance of Equation 3, we obtain the desired function $\pi^* : C \to \Gamma$ by setting $\pi^*(a)$ to be the (unique) element $k \in \Gamma$ satisfying $u_k(a) = 1$, and a decomposition $\mathsf{conf}(\pi^*) = \sum_{i=1}^{\alpha} \mathbf{v}_i$ into legal vectors. As argued before, we can then use Lemma 5 to compute an optimal solution for the problem in $O(n)$ time. The overall running time of our algorithm is $2^{2^{O(d)}} n^{O(1)}$. This proves Theorem 1, which we restate below.

▶ **Theorem 1.** *There exists a $2^{2^{O(d)}} n^{O(1)}$-time algorithm for* Multi-team Formation.

Although the running time of our algorithm depends double exponentially on d, it is ETH-tight and hence unlikely to be substantially improved. The lower bound follows readily from the reduction in [20] and the ETH lower bound in [1] for 3-dimensional Matching.

▶ **Theorem 2.** *The existence of a $2^{2^{d/c}} n^{O(1)}$-time algorithm for* Multi-team Formation *with any constant $c > 12$ violates the Exponential Time Hypothesis (ETH).*

Proof. Let $c > 12$ be a constant. Schibler et al. [20] described a polynomial-time reduction from 3-dimensional Matching to Multi-team Formation with $n = O(m)$ and $d = 12\log m + O(1)$, where m is the size of the 3-dimensional Matching instance. Therefore, a $2^{2^{d/c}} n^{O(1)}$-time algorithm for Multi-team Formation implies a $2^{m^{12/c}} m^{O(1)}$-time algorithm for 3-dimensional Matching. However, it was shown in [1] that any algorithm with running time $2^{o(m)}$ for 3-dimensional Matching violates the ETH. ◀

3 Conclusion and future work

In this paper, we considered Multi-team Formation under the natural sum-of-maxima scoring rule, and presented an algorithm that runs in $2^{2^{O(d)}} \cdot n^{O(1)}$ time, which is ETH-tight since a $2^{2^{d/c}} \cdot n^{O(1)}$-time algorithm, for any constant $c > 12$, would violate the ETH.

A direction for future work is approximation algorithms for Multi-team Formation. Exploiting the submodularity of the sum-of-maxima scoring function, one can easily formulate Multi-team Formation as a submodular maximization problem with two matroid constraints, which leads to a polynomial-time $(0.5-\varepsilon)$-approximation algorithm for any constant $\varepsilon > 0$ using the algorithm of [13]. Whether one can achieve a better approximation in polynomial time is an interesting open question to be studied.

References

1 Nikhil Bansal, Tim Oosterwijk, Tjark Vredeveld, and Ruben Van Der Zwaan. Approximating vector scheduling: almost matching upper and lower bounds. *Algorithmica*, 76(4):1077–1096, 2016.

2 Marek Cygan, Marcin Pilipczuk, and Michal Pilipczuk. Known algorithms for edge clique cover are probably optimal. *SIAM J. Comput.*, 45(1):67–83, 2016.

3 Friedrich Eisenbrand, Christoph Hunkenschröder, Kim-Manuel Klein, Martin Koutecký, Asaf Levin, and Shmuel Onn. An algorithmic theory of integer programming. *arXiv preprint*, 2019. `arXiv:1904.01361`.

4 Uriel Feige. A threshold of $\ln n$ for approximating set cover. *J. ACM*, 45(4):634–652, 1998.

5 Erin L. Fitzpatrick and Ronald G. Askin. Forming effective worker teams with multi-functional skill requirements. *Computers & Industrial Engineering*, 48(3):593–608, 2005.

6 Raymond Hemmecke, Shmuel Onn, and Lyubov Romanchuk. N-fold integer programming in cubic time. *Mathematical Programming*, 137(1-2):325–341, 2013.

7 Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972.

8 Jon Kleinberg and Maithra Raghu. Team performance with test scores. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, EC '15, pages 511–528, 2015.

9 Dušan Knop and Martin Koutecký. Scheduling meets n-fold integer programming. *Journal of Scheduling*, 21(5):493–503, 2018.

10 Martin Koutecký, Asaf Levin, and Shmuel Onn. A parameterized strongly polynomial algorithm for block structured integer programs. In *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

11 Theodoros Lappas, Kun Liu, and Evimaria Terzi. Finding a team of experts in social networks. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 467–476, 2009.

12 Patrick R. Laughlin and Andrea B. Hollingshead. A theory of collective induction. *Organizational Behavior and Human Decision Processes*, 61(1):94–107, 1995.

13 Jon Lee, Maxim Sviridenko, and Jan Vondrák. Submodular maximization over multiple matroids via generalized exchange properties. *Mathematics of Operations Research*, 35(4):795–806, 2010.

14 Tomasz P. Michalak, Talal Rahwan, Edith Elkind, Michael J. Wooldridge, and Nicholas R. Jennings. A hybrid exact algorithm for complete set partitioning. *Artif. Intell.*, 230:14–50, 2016.

15 George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. An analysis of approximations for maximizing submodular set functions - I. *Math. Program.*, 14(1):265–294, 1978.

16 Scott Page. *The Difference: How the Power of Diversity Creates Better Groups, Firms, Schools, and Societies.* Princeton University Press, 2007.

17 Marcin Pilipczuk and Manuel Sorge. A double exponential lower bound for the distinct vectors problem. *CoRR*, abs/2002.01293, 2020. `arXiv:2002.01293`.

18 Habibur Rahman, Senjuti Basu Roy, Saravanan Thirumuruganathan, Sihem Amer-Yahia, and Gautam Das. Optimized group formation for solving collaborative tasks. *The VLDB Journal*, 28(1):1–23, February 2019.

19 Talal Rahwan, Tomasz P. Michalak, Michael J. Wooldridge, and Nicholas R. Jennings. Coalition structure generation: A survey. *Artif. Intell.*, 229:139–174, 2015.

20 Thomas Schibler, Ambuj Singh, and Subhash Suri. On multi-dimensional team formation. In *Proc. of the 31st Canadian Conference on Computational Geometry*, pages 146–152, 2019.

21 Travis C. Service and Julie A. Adams. Coalition formation for task allocation: theory and algorithms. *Autonomous Agents and Multi-Agent Systems*, 22(2):225–248, March 2011.

22 Marjorie E. Shaw. A comparison of individuals and small groups in the rational solution of complex problems. *The American Journal of Psychology*, 44(3):491–504, 1932.

23 Onn Shehory and Sarit Kraus. Methods for task allocation via agent coalition formation. *Artif. Intell.*, 101(1-2):165–200, 1998.

24 I. D. Steiner. *Group process and productivity.* New York: Academic Press, 1972.

25 Xinyu Wang, Zhou Zhao, and Wilfred Ng. A comparative study of team formation in social networks. In *Database Systems for Advanced Applications - 20th International Conference, DASFAA 2015*, pages 389–404. Springer, 2015.