On Multi-Dimensional Team Formation

Thomas Schibler[*] Ambuj Singh[†] Subhash Suri[‡]

Abstract

We consider a team formation problem in multi-dimensional space where the goal is to group a set of n agents into α teams, each of size β, to maximize their total performance. The performance of each team is measured by a score, which is the sum of h highest skill values in each dimension. We wish to maximize the sum of team scores. We prove that the problem is NP-hard if the dimension is $d = \Omega(\log n)$ even for $h = 1$ and $\beta = 4$. We then describe an efficient algorithm for solving the problem in two dimensions as well an algorithm for computing a single optimal team in any constant dimension.

1 Introduction

The problem of grouping a set of agents into teams with the objective of optimizing their collective performance is ubiquitous in a variety of organization settings, including team sports, project management, law, military, management consulting, academic ad hoc committees, to name a few. Mathematical models of team selection and performance, therefore, are an important area of research in social and management sciences. In these models, the skill set of each individual is typically modeled as an attribute vector. Research shows that while individual skills are clearly an important factor, the team's ability to search over vast and often ill-defined decision space crucially depends on its overall synergy and diversity [7, 14, 18, 20]. As a result, it is widely recognized that the performance of a team along a specific skill dimension should not depend on the average of the group members' values (so called weak synergy [6]) but rather on the skills of the best individuals on each dimension [1, 10, 19].

The selection of a single best team has been considered broadly in the literature [2, 3, 12, 13, 9, 11, 4, 15]. The more general problem of assembling multiple teams, however, is less well-understood, and has been studied mainly in the context of very specific performance objectives. For instance, Fitzpatrick and Askin [5] develop heuristics for assembling multiple 'multi-functional' teams using an integer programming formulation. The *coalition formation* problem in multi-agent (and multi-robot) systems also partitions agents into teams but the primary goal there is strategic utility maximization of completing a given set of tasks [16]. When the utility function is a simple sum of scalars, this becomes an easy-to-solve assignment problem in bipartite graphs [17], but under arbitrary set-valued functions the coalition formation is both NP-hard and inapproximable [16].

Against this backdrop, in this paper we investigate a simple and natural model for assembling multiple teams with multi-dimensional skills that allows us to explore the computational complexity of multi-team formation as a function of the problem's intrinsic parameters: number of agents n, number of teams α, team size β, and dimension d of the skill vector. We place no constraints on the team structure except its prescribed size—any subset of agents can form a team—and use a simple additive function over independent attributes to measure team performance, thereby isolating the combinatorial aspects of the problem.

Specifically, we have an agent pool of n candidates, each modeled as a d-dimensional point $\mathbf{p} = (p_1, p_2, \ldots, p_d)$, where each dimension represents an independent real-valued skill. We want to form α teams, each of size β, for some integer values α, β, with $\alpha\beta \leq n$, so as to *maximize* the total score of all the teams. Each agent belongs to at most one team. In formulating the team score, we combine the two important aspects of a team performance: *strength* and *robustness* [3]. We measure the team strength by its coordinate-wise maxima but in order to add some degree of robustness we take the *top h* values for each coordinate, for a user-specified parameter $h \geq 1$. Thus, a team's score is defined as the *sum of h highest values of all dimensions.* We use the notation $\text{score}_h(T)$ to denote the score of team T using the top h scoring rule, which can be formally defined as

$$\text{score}_h(T) = \sum_{j=1}^{d} \max_{S \subset T, |S|=h} \sum_{\mathbf{p}_i \in S} p_j^i,$$

where p_j^i is the jth coordinate of the ith point $\mathbf{p}_i$. Our problem then is the following: given a set of n agents, form α teams $T_1, T_2, \ldots, T_\alpha$, each of size β to maximize $\sum_i \text{score}_h(T_i)$. Figure 1 shows an example in two di-

[*]Computer Science Department, University of California, Santa Barbara, CA 93106, USA, tschibler@gmail.com

[†]Computer Science Department, University of California, Santa Barbara, CA 93106, USA, ambuj@cs.ucsb.edu

[‡]Computer Science Department, University of California, Santa Barbara, CA 93106, USA, suri@cs.ucsb.edu

mensions, where (A, C, D) is an optimal team of size 3 for the instance on the left using scoring parameter $h = 2$.

Our Results

We show that the multi-team formation problem is NP-hard for dimension $d = \Omega(\log n)$, even with $h = 1$ and $\beta = 4$. Specifically, we reduce the well-known NP-complete problem of 3-Dimensional Matching to the team formation problem in dimension $d = \Omega(\log n)$. (If we consider very large dimensions, namely, $d = \Omega(n)$, the problem becomes trivially hard because simply acquiring all necessary skills is a set covering problem. In most realistic settings, however, the dimension is much smaller than n, which is the focus of our work.) Our main result is a polynomial time algorithm for solving the 2-dimensional team formation problem optimally, for all scoring rules $h \geq 1$, using the following two-step algorithm. We first form a single team of size $\alpha \times \beta$, which we call a *league*, using a modified scoring rule. We prove that the total score of the league equals the score of the optimal team formation, and that an optimal league can be decomposed into an optimal solution of the team formation in polynomial time.

Our dynamic programming based algorithm can compute an optimal league in any fixed dimension. However, we show that a key structural result, called *league decomposition* lemma, fails in higher dimensions, and so the optimal league's score no longer equals the score of the optimal team formation problem. Thus, forming multiple teams in more than two, but a constant, dimension remains an open problem.

2 Hardness of Team Formation

We begin with a brief reintroduction of the multi-team formation problem. Given an agent pool of n candidates, each candidate modeled as a d-dimensional point $\mathbf{p} = (p_1, p_2, \ldots, p_d)$, we want to form α teams, each of size β, for some integer values α, β, with $\alpha\beta \leq n$, so as to *maximize the sum* of team scores. Each agent belongs to at most team, and the score of a team T is defined as

$$\text{score}_h(T) \quad = \quad \sum_{j=1}^{d} \max_{S \subset T, |S|=h} \sum_{\mathbf{p}_i \in S} p_j^i,$$

where p_j^i is the jth coordinate of the ith point $\mathbf{p}_i$, and $h \geq 1$ is the scoring parameter.

Theorem 1 *The multi-team formation problem is NP-hard.*

Proof. We reduce the well-known 3-dimensional matching (3DM) [8] problem to our problem. An instance of 3DM consists of three input sets X, Y, Z each of size n and a set of triples $W \subset X \times Y \times Z$. The problem is to decide if there exists a subset of n triples $T \subseteq W$ so that each element of $X \cup Y \cup Z$ is contained in exactly one of the triples.

Given an instance of 3DM, we create an instance of the team formation problem as follows. The number of dimensions in our problem will be $6\ell+4$, for a parameter $\ell = \Theta(\log n)$. For each element of $X \cup Y \cup Z$, we associate a unique bit string of length 2ℓ, containing ℓ zeros and ℓ ones. We call this string the *tag* of that element. The bitwise complement of a tag t will be denoted t'. In particular, we will use the following types of bit strings:

1. $\text{tag}(x) \;=\;$ a unique bit string of length 2ℓ containing ℓ bits of 0 and ℓ bits of 1
2. $\text{tag}'(x) \;=\;$ bitwise complement of $\text{tag}(x)$
3. $0_\ell \;=\;$ a string of length ℓ containing all 0

Using the fact that $\binom{2\ell}{\ell} \geq 2^\ell$, the choice of $\ell = 2\log n$ suffices for the creation of $3n$ distinct tags, one for each element of $X \cup Y \cup Z$. With the help of these tags we now create a point for each element of $X \cup Y \cup Z$ and each triple $t = (u, v, w)$ of W, in dimension $6\ell + 4$, as follows:

$x:$	$\text{tag}(x)$	$0_{2\ell}$	$0_{2\ell}$	1	0	0	0
$y:$	$0_{2\ell}$	$\text{tag}(y)$	$0_{2\ell}$	0	1	0	0
$z:$	$0_{2\ell}$	$0_{2\ell}$	$\text{tag}(z)$	0	0	1	0
$t:$	$\text{tag}'(u)$	$\text{tag}'(v)$	$\text{tag}'(w)$	0	0	0	1

Specifically, the first 2ℓ dimensions of $x \in X$ are its tag bits, followed by 4ℓ bits of 0's, and its last four bits are 1 0 0 0. The patterns for $y \in Y$ and $z \in Z$ are similar, as shown above. Next, the point corresponding to a triple $t = (u, v, w)$ has 6ℓ bits corresponding to the *tag'* strings of u, v, w, followed by the pattern 0 0 0 1. Altogether we have $3n + |W|$ points in dimension $O(\log n)$, which is polynomial in the input size.

We now prove the following: the input 3DM instance is a yes instance if and only if our constructed instance admits formation of $\alpha = n$ teams, each of size $\beta = 4$, with total score at least $n(6\ell + 4)$. To prove the forward direction, suppose the 3DM instance has a solution given by the set of triples T. For each $t = \{x, y, z\} \in T$, we create a team of size 4 using the points corresponding to x, y, z and t. Since $|T| = n$, and no element appears in more than one triple, we can form n disjoint teams. We now show that these teams achieve the target total score.

Each point's coordinate is either 0 or 1 along each of the $6\ell + 4$ dimensions, and so to reach the target score, each team must collect a 1 in each dimension, using its four points. Suppose the four points correspond to x, y, z and the triple (x, y, z). Then, by construction, in each of the first 6ℓ dimensions, we have a 1 in either tag() or tag$'$(), satisfying the requirement. Finally, the same holds for the last four dimensions, which is easy to check by inspection. Thus, assuming that the 3DM instance has a satisfying solution, we can construct n teams, each of size 4 with total score $n(6\ell + 4)$.

In the reverse direction, we show that any set of n teams with this score correspond to a perfect 3 dimensional matching. First, we observe that the optimal score requires that every team contributes exactly one 1 in each dimension. Considering the last 4 dimensions alone, this is only possible if the team contains exactly one point corresponding to a triple and each of the 3 elements in X, Y, and Z. Given this team structure, each of the first 3 sets of 2ℓ dimensions must collect a 1 from either the tag or tag$'$ of some element or triple respectively. To satisfy all 2ℓ dimensions, the tag and the tag$'$ must correspond to the same element, otherwise they will not be bitwise complements of each other. Consequently, we must have elements x, y, z and triple $t = \{u, v, w\}$ with $x = u$, $y = v$, and $z = w$. If this property holds for all teams, then all selected triples must exactly cover each of the element sets, proving the existence of a 3DM solution. This completes the proof. □

The hardness proof is easily extended to any team size $\beta \geq 4$ by introducing an appropriate number of agents with null skills, namely, points with all 0 coordinates. The argument requires increasing the number of dimensions by $\Omega(\beta)$ to avoid the use of multiple triples in a single team. If the dimension is $\Omega(n)$, then computing a *single* team is also intractable. The proof can also be extended to scoring rule with $h > 1$ by introducing an appropriate number of points whose all coordinates are 1s.

3 An Efficient Algorithm for 2 Dimensions

Given the NP-hardness of the general problem, we now consider optimal team formation in small dimensions. In one dimension, the problem can be easily solved in $O(n \log n)$ time, as follows. We sort the agents in the increasing order of the skill level, say, the x axis. We then repeatedly select the top h unassigned agents, and assign them to the next team, until each of the α teams has h agents. Clearly, this assignment has the maximum sum of team scores. If needed, we can make each team's size to be exactly $\beta \geq h$, by arbitrarily selecting any of the unassigned agents since their scores do not contribute to the team scores.

In fact, a similar greedy strategy also solves the team formation problem for any dimension $d \geq 1$ if the team size is $\beta \geq hd$: repeat the earlier one-dimensional algorithm independently for each dimension. (It is possible for an agent to contribute a top score in more than one dimension, in which case a team may reach its maximum possible score with fewer than hd agents.) For team size $\beta < hd$, however, the problem becomes nontrivial even in dimension $d = 2$ and $h = 2$. This is the focus of the following discussion, where we consider the team formation problem in two dimensions.

3.1 Forming 3-person teams in 2-dimensions

In the interest of simpler exposition and proofs, we describe our algorithm using the scoring rule $h = 2$, and then discuss the minor adaptations needed for generalization to higher values of h. Therefore, in the following we drop the subscript h from the scoring notation; it is always assumed to be $h = 2$. Specifically, we focus on the case of team size $\beta = 3$, which helps illustrate some of the main difficulties of the problem. The case of $\beta = 1$ or $\beta = 2$ is easily solved greedily in two dimensions, and thus omitted from our discussion.

Somewhat surprisingly, the problem of forming teams of size 3 turns out to be non-trivial even if we want to form a single team, namely, $\alpha = 1$ with the scoring parameter $h = 2$. It serves as a test case both for disproving greedy schemes, and for our polynomial time algorithm. Using x and y as coordinates in two dimensions, suppose a 3-person team has agents with coordinates $(x_1, y_1), (x_2, y_2), (x_3, y_3)$. (Recall that we are using scoring rule of top two values, namely, $h = 2$.) Since the team score is composed of top two x and top two y values, and there are only 3 agents, at least one of them contributes both x and y to the team score. Let us call such an agent a 2-contributor. Each of the other agents contributes its x or y values (possibly these two agents are the same).

This property of the optimal 3-person team suggests a natural greedy algorithm: sort the agents by their x, y, and $x + y$ values. First take the agent with the maximum $x + y$, remove it from all three lists, and then choose the agents with the maximum x and maximum y. Unfortunately, this simple algorithm is flawed. In fact, one can show that any algorithm that selects the team using only the *rank order* by x, y and $x + y$ coordinates fails. In particular, we construct two instances, each containing 4 agents, whose sorted orders by x, y, and $x+y$ are identical, yet their top scoring teams are different. The construction is shown in Figure 1.

On the left, we have an instance with four agents $A = (4, 11), B = (5, 5), C = (1, 8), D = (8, 1)$. The optimal 3-person team for this instance is (A, C, D) with score of $31 = x(A) + x(D) + y(A) + y(C)$, with A contributing both x and y, C contributing y and D contributing x.

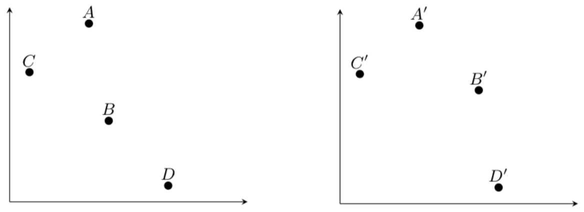

Figure 1: An example for $\alpha = 1, \beta = 3, h = 1$ and $d = 2$. On the left, we show an instance of team formation with four two-dimensional agents: $A = (4, 11), B = (5, 5), C = (1, 8), D = (8, 1)$. On the right, we show a closely related instance with $A' = (4, 11), B' = (7, 7), C' = (1, 8), D' = (8, 1)$. The two instances have exactly the same sorted orders along $x, y, x+y$, but they lead to different optimal 3-person team solutions. The optimal team for the left instance is (A, C, D) with score 31 while the team for the right instance is (A', B', D') with score 33.

On the right, we have another instance also with four agents, where only the coordinates of B' are different: $A' = (4, 11), B' = (7, 7), C' = (1, 8), D' = (8, 1)$, whose optimal team is (A', B', D') with score of $33 = x(B') + x(D') + y(A') + y(B')$. Yet, the two instances have the same ranking order by x, y, and $x+y$. *The crucial point of this example is that although A is an obvious choice for inclusion in the team, whether it contributes both x and y or just y depends on which other agents are in the team, namely, B or B'.*

Of course, since there are only $O(n^3)$ choices for a 3-person team, one can exhaustively find an optimal one. But what about forming α teams? Even for the simple sum-of-team-scores objective function, the greedy strategy of iteratively computing the best 3-person team among the remaining agents fails, as shown by the following example of six agents that we want to group into two teams of size 3.

$$A = (20, 20); B = (10, 20); C = (20, 10);$$

$$D = E = F = (0, 0)$$

The single optimal team is (A, B, C), with an score of 80, which leaves the remaining team of (D, E, F) with score 0. Instead, an optimal choice of two teams would be (A, B, D) and (C, E, F), which together have a score of 100.

3.2 A Polynomial Time Algorithm

In the following, we develop a polynomial time algorithm for solving the multi-team formation problem optimally in two dimensions. Our algorithm is based on the following idea:

1. First, identify the *union* of all the agents that are in the optimal set of teams, and then
2. Partition this union into individual teams while preserving the total score.

A 3-person team in dimension $d = 2$ involves a total of $2d = 4$ individual skill scores, namely, top two scores in each of the two dimensions. Across α teams, therefore, we have a total of 4α scores. Instead of forming these teams, let us consider a slightly different problem. Find a group of 3α agents whose score is computed as follows: for each dimension, we take the top 2α skill values, and the group score is the sum of these 4α values.

For ease of reference, let us call such a group of 3α agents with this new scoring rule a *league*. Given a league L, let score(L) be the total score of L. Suppose $\mathcal{T}$ is the optimal set of 3-person teams, with total score score$(\mathcal{T})$. The question we ask is: what is the gap between score$(\mathcal{T})$ and score(L)? Clearly, score$(\mathcal{T}) \leq$ score(L), because the union of $\mathcal{T}$ is a valid league: a group of 3α agents, whose dimension-wise scores add up to score(L). But how much larger can the league score be compared to the team score? Our main result is that the two are always equal in two dimensions and, more importantly, (1) an optimal league can be partitioned in polynomial time into α teams of size $\beta = 3$ with the same total score, and (2) we can compute an optimal league in polynomial time. Together the two lead to a polynomial time algorithm.

3.3 Optimal League Decomposition

Let us first establish the league decomposition lemma.

Lemma 2 (League Decomposition) *Given an instance of multi-team formation in two dimensions, let $\mathcal{T}$ be an optimal solution of α teams of size 3 each, and let L be an optimal league of size 3α. Then,* score$(\mathcal{T})$ = score(L). *We can also partition L into an optimal multi-team solution in time $O(n)$.*

Proof. The score of a league sums the top 2α values in each dimension. We label each point a 2-contributor, x-contributor, y-contributor, or none, depending on how many coordinate values it contributes to the league score. We then observe the following:

1. there are at least α 2-contributors.
2. there are an equal number of x and y-contributors, and this number is at most α.

The first claim follows from the pigeon hole principle: 4α values are summed in scoring the league, but there are only 3α points, and so at least α points must contribute both of their coordinates. This leaves at most 2α

values unaccounted for, which must be evenly split between x and y values. Thus, at most α values can come from an x-contributor, and α from a y-contributor.

```
1: procedure PARTITION 2D LEAGUE($\mathbf{p}_1, \cdots, \mathbf{p}_{3\alpha}$)
2:     Initialize contributor lists $X$, $Y$, and $XY$.
3:     Initialize empty list of teams $\mathcal{T}$.
4:     for $i \in [0, \mathbf{len}(X)]$ do
5:         Add team $\{XY[i], X[i], Y[i]\}$ to $\mathcal{T}$.
       $j \leftarrow \mathbf{len}(X)$.
6:     while $j < \mathbf{len}(XY)$ do
7:         Select any unused point $\mathbf{p}$.
8:         Add team $\{XY[j], XY[j+1], \mathbf{p}\}$ to $\mathcal{T}$.
9:         $j \leftarrow j + 2$.
10:    Return $\mathcal{T}$.
```

Figure 2: Partitioning a league into optimal teams.

We use these two facts to design a simple greedy algorithm for partitioning the league. The algorithm is shown above in Fig. 2. We first pair any 2-contributor of the league with one x-contributor and one y-contributor. Because there are at least as many 2-contributors as x or y-contributors, we can continue this until there are no more 1-contributors left. By (2) above, we exhaust the x and y contributors at the same time. If any 2-contributors remain, we pair them arbitrarily together, along with an arbitrary extra point if we wish to maintain the team size.

To see that the resulting teams have the same total score as L, we note that exactly two x and two y values contributing to the league score are assigned to each team. Finally, the greedy algorithm only uses unsorted lists, and therefore runs in $O(n)$ time. This completes the proof. $\square$

3.4 Computing an Optimal League

We now describe an algorithm for computing the optimal league L, using dynamic programming. Given a set of n d-dimensional points $\mathbf{p}_1, \cdots, \mathbf{p}_n$, we construct a 4-dimensional table A of size $n \times 3\alpha \times 2\alpha \times 2\alpha$ whose $A[i, j, k, l]$ entry stores $\text{score}_{k,l}(L_{i,j})$, where

$$\begin{aligned} L_{i,j} &= \text{an optimal league using at most} \\ &\quad\ j \text{ points in } \{\mathbf{p}_1, \cdots, \mathbf{p}_i\} \\ \text{score}_{k,l}(L) &= \text{sum of top } k \text{ x-values and top } l \\ &\quad\ \text{y-values of } L \end{aligned}$$

The table is initialized as $L_{0,j} = L_{i,0} = 0$, for all i, j. Suppose we have computed all $L_{i-1,j-1}$ and want to compute $L_{i,j}$. Consider the new point $\mathbf{p}_i$. It is either not included in the league, or if it is included it serves in one of the three possible roles: x-contributor, y-contributor, or 2-contributor. We can, therefore, compute the table entry $L_{i,j}$ using the following dynamic program:

```
1: procedure 2D LEAGUE($\mathbf{p}_1, \cdots, \mathbf{p}_n$, $\alpha$)
2:     Initialize $n \times 3\alpha \times 2\alpha \times 2\alpha$ table $A$
3:     Let $A[0, j, k, l] = 0,\ \forall j, k, l$
4:     Let $A[i, 0, k, l] = 0,\ \forall i, k, l$
5:     for $i \in [1, n]$ do
6:         for $j \in [1, 3\alpha]$ and $k, l \in [0, 2\alpha]$ do
7:             $s_x \leftarrow A[i-1, j-1, k-1, l] + \mathbf{p}_i[x]$
8:             $s_y \leftarrow A[i-1, j-1, k, l-1] + \mathbf{p}_i[y]$
9:             $s_{x,y} \leftarrow A[i-1, j-1, k-1, l-1] + \mathbf{p}_i[x] + \mathbf{p}_i[y]$
10:            $s_0 \leftarrow A[i-1, j, k, l]$
11:            $A[i, j, k, l] \leftarrow \max(s_x, s_y, s_{x,y}, s_0)$
12:    Return $A[n, 3\alpha, 2\alpha, 2\alpha]$
```

Specifically, if $\mathbf{p}_i$ is an x-contributor, then $\text{score}_{k,l}(L_{i,j})$ is the x-coordinate of $\mathbf{p}_i$ plus the $\text{score}_{k-1,l}(L_{i-1,j-1})$; that is, the remaining points may only contribute $k-1$ x-values. We have similar cases for $\mathbf{p}_i$ being a y-contributor or a 2-contributor. The final optimal league score is found in the table entry $A[n, 3\alpha, 2\alpha, 2\alpha]$.

The table A has size $O(n\alpha^3)$, each entry can be computed in constant time, and so the algorithm runs in $O(n\alpha^3)$ time and space.

3.5 Extension to Top h Scoring Rule

The league decomposition lemma and the algorithm for computing the optimal league easily extend to scoring rule of top h, for all $h \geq 2$, as follows. Without loss of generality, we may assume that the team size satisfies $h \leq \beta < 2h$. Thus, the league size satisfies $\alpha\beta < 2\alpha h$. By the pigeonhole principle, the number of 2-contributors in the league is at least $\alpha(2h - \beta)$, and therefore we can assign to each team at least $(2h-\beta)$ of these 2-contributors, and fill the rest by 1-contributors arbitrarily. Similarly, the dynamic program algorithm is easily extended by changing the table size to $h\alpha$ instead of 2α. We summarize the main result of our paper.

Theorem 3 *The multi-team formation problem in two dimensions can be solved optimally in worst-case time and space $O(n\alpha^3\beta h^2)$, where α is the number of teams, β the team size, h the scoring parameter, and n is the number of agents.*

4 Team Formation in Higher Dimensions

The dynamic programming algorithm of Section 3.3 can be extended to form an optimal *single* team of size

$\beta < hd$ in polynomial time, for any fixed dimension d. Specifically, we compute a $(d+2)$-dimensional table A, whose first two dimensions are the same as before, namely, the first i points and the team size j. Each of the remaining d indices corresponds to the number of top scores in each dimension. In particular, $\text{score}_{k_1,\cdots k_d}$, where each $k_i \in [0, h]$, is the team score where top k_i values in dimension i have been accounted for. There are 2^d such combinations, so each table entry can be computed in $O(2^d)$ time. As mentioned earlier, when the team size $\beta \geq hd$, the problem can be easily solved in $O(dn)$ time using a greedy algorithm. We therefore have the following result.

Theorem 4 *We can compute an optimal single team of size β in d dimensions in time $O((2h)^d \beta n)$ time.*

The real difficulty in higher dimensions lies in forming *multiple teams*. In two dimensions, we used the League Decomposition Lemma as a key tool. Unfortunately, as we show below, in higher dimensions, this lemma no longer holds.

Theorem 5 *Let L be an optimal league, and $\mathcal{T}$ a set of optimal teams in dimensions $d \geq 4$. Then, there are instances for which* $\text{score}(\mathcal{T}) < \text{score}(L)$.

Proof. Consider the following set of 9 agents in four dimensions. $A = A' = (1,1,1,0), B = B' = (1,1,0,1), C = C' = (1,0,1,1), D = D' = (0,1,1,1)$, and $F = (0,0,0,0)$. Suppose our goal is to form $\alpha = 3$ teams, each of size $\beta = 3$. Then, trivially, our league consists of all p points, where the scoring rule sums the top $2\alpha = 6$ values in each dimension. By construction, we have six 1s in each dimension, and so $\text{score}(L) = 24$.

However, any partition of these nine agents into 3 teams must assign the all-zero point F to one of the teams, which can therefore have a score of at most 6. On the other hand, no team has score more than 8, since the sum of top two entries in each of the four dimensions is two. Thus, the optimal team formation has score 22, proving that $\text{score}(\mathcal{T}) < \text{score}(L)$. This completes the proof. $\square$

One can also show that an optimal partition of an optimal league L may not give an optimal team formation solution $\mathcal{T}$. For instance, imagine introducing one more agent $G = (1,1,1,1)$ to the set of points in the previous example. Replacing F by G does not improve $\text{score}(L)$, so L remains an optimal league. On the other hand, replacing F by G does improve $\text{score}(\mathcal{T})$. Finding an efficient algorithm for optimal or approximately optimal team formation in a constant dimension larger than 2 remains an interesting open problem.

5 Concluding Remarks

Our work introduces a simple and natural model for multi-dimensional multi-team formation, and shows that computing optimal teams is NP-hard even in moderate dimensions. We show that the problem of forming multiple teams optimally can be efficiently solved in two dimensions, as is the problem of forming a single team in any dimension $d = O(\log n)$. The problem of forming multiple teams in higher than two dimension, either exactly or approximately, remains an interesting open problem.

There are several other natural objective functions for team optimization, such as maximizing the minimum team score, instead of maximizing the sum of team scores. For maximizing the minimum, unfortunately, we can show that the problem is NP-hard even in one dimension if we sum the top three scores of the team. The objective function can also be extended by considering other aspects of team formation that translate to more general constraints beyond individual-specific skills: for example, synergy between team members translates to edge-level requirements. Learning the skills and synergies based on past observations is another possible future extension of the research.

References

[1] Bryan L. Bonner, Michael R. Baumann, Austin K. Lehn, Daisy M. Pierce, and Erin C. Wheeler. Modeling collective choice: decision-making on complex intellective tasks. *European Journal of Social Psychology*, 36(5):617–633, 2006.

[2] Shi-Jie Chen and Li Lin. Modeling team member characteristics for the formation of a multifunctional team in concurrent engineering. *IEEE Transactions on Engineering Management*, 51(2):111–124, 2004.

[3] Chad Crawford, Zenefa Rahaman, and Sandip Sen. Evaluating the efficiency of robust team formation algorithms. In *Autonomous Agents and Multiagent Systems*, pages 14–29, Cham, 2016. Springer International Publishing.

[4] Christoph Dorn and Schahram Dustdar. Composing near-optimal expert teams: A trade-off between skills and connectivity. In *On the Move to Meaningful Internet Systems: OTM 2010*, pages 472–489, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[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] Larson J.R., Jr. *In search of synergy: In small group performance.* Psychology Press, Taylor & Francis, New York, 2010.

[7] 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.

[8] Jon M. Kleinberg and Éva Tardos. *Algorithm design.* Addison-Wesley, 2006.

[9] 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.

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

[11] C. Li and M. Shan. Team formation for generalized tasks in expertise social networks. In *2010 IEEE Second International Conference on Social Computing*, pages 9–16, Aug 2010.

[12] Somchaya Liemhetcharat and Manuela Veloso. Modeling and learning synergy for team formation with heterogeneous agents. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '12, 2012.

[13] Somchaya Liemhetcharat and Manuela Veloso. Weighted synergy graphs for effective team formation with heterogeneous ad hoc agents. *Artificial Intelligence*, 208:41 – 65, 2014.

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

[15] 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.

[16] Tuomas Sandholm, Kate Larson, Martin Andersson, Onn Shehory, and Fernando Tohmé. Coalition structure generation with worst case guarantees. *Artif. Intell.*, 111(1-2):209–238, 1999.

[17] 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, Mar 2011.

[18] 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.

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

[20] Anita Williams Woolley, Christopher F. Chabris, Alex Pentland, Nada Hashmi, and Thomas W. Malone. Evidence for a collective intelligence factor in the performance of human groups. *Science*, 330(6004):686–688, 2010.