Submodularity in Team Formation Problem*

Avradeep Bhowmik† Vivek Borkar‡ Dinesh Garg§ Madhavan Pallan¶

Abstract

We consider the team formation problem where the goal is to find a team of experts for a specific project. In the past, several attempts have been made to formulate this problem and each formulation focuses only on a subset of design criteria such as skill coverage, social compatibility, economy, skill redundancy, etc. In this paper, for the first time, we show that most of the important design criteria for this problem can be fully modeled within one single formulation as an *unconstrained submodular function maximization* problem. In our formulation, the submodular function turns out to be non-negative and non-monotone. The maximization of this class of submodular function is much less explored than its *monotone constrained* counterpart. A few recent works [7] [4] have come up with a simulated annealing based randomized approximation scheme for this problem with an approximation ratio of 0.41. In this paper, we customize this algorithm to our formulation and conduct an extensive set of experiments to show its efficacy. Our proposed formulation offers several advantageous features over the existing formulations including *skill cover softening*, *better team communication*, and *connectivity relaxation*. Unlike previous formulations, the skill cover softening feature allows a designer to specify *Must Have* and *Should Have* skills. Similarly, through better team communication, we avoid the restriction that all the communication among team members should pass through only the team members and not the outsiders. Finally, connectivity relaxation feature alleviates the constraint of whole team being connected and thereby, lowering the cost.

1 Introduction

A commonly occurring problem faced by any organization that works with skilled human resources is to put together a team of experts for a specific project. This problem is generally referred to as the *team formation problem*. An example of this could be a software firm putting together a team of software engineers for the development of a specific product. The success of the project depends on multiple factors which serve as design criteria for the team formation problem like *skill coverage*, *social compatibility*, and *economy*. That is, the team must be formed in a way that the team members collectively possess the required skills for the project and the team members are compatible with each other and can work well in a team. Also, it doesn't help making the team larger than necessary, or including redundant members in the team- the team should be as small as possible while still satisfying the necessary constraints. There could be other secondary design criteria too while forming the team, like load balancing across team members, ensuring the presence of some specific members in the team, etc.

The purpose of this paper is to advance the state-of-the-art in the team formation problem by showing that *this problem exhibits a very natural submodular structure and most of the design criteria can be modeled within one single formulation, which is unconstrained submodular function maximization* - an NP-hard problem for arbitrary submodular functions. We show that this submodular formulation is quite generic and most of the existing formulations turn out to be special cases of this. Further, this new formulation offers several advantages and extensions over and above the existing formulations and thereby, advances our understanding about this problem.

1.1 Prior Art

The team formation problem has been studied since the mid 2000s in the Operations Research community [3, 15, 2, 14]. In early works, the problem used to be modeled as purely an assignment or matching problem, where the goal was to match people with required skills. The solution methodology for these formulations spanned across simulated annealing [2], branch-and-cut [15], and genetic algorithms [14].

As mentioned earlier, this problem has an obvious and natural social aspect but this aspect was paid no attention to until very recently when Lappas et al [10] argued that apart from ensuring the required skills set, it is essential to take into account the social well-being of the team which refers to the ease of communication and willingness to collaborate among team members. The authors in [10] proposed the idea of quantifying social well-being of a team by means of a communication cost among the team members which is typically evaluated from a given social graph whose edge weights represent

*This work was supported by IBM SUR grant.

†University of Texas, Austin, avradeep.1@utexas.edu
‡IIT Bombay, India, borkar@ee.iitb.ac.in
§IBM Research, India, garg.dinesh@in.ibm.com
¶IBM Research, India, mapallan@in.ibm.com

the ease (or difficulty) of communication among the team members. The authors in [10] defined two different cost criteria - the diameter and the minimum spanning tree (MST) on the subgraph of selected team members. Approximation algorithms were proposed to find a team of experts possessing a given set of skills while minimizing these cost criteria.

Many extensions of [10] have surfaced over the past few years. Kargar et al [8] pointed out a few limitations in the cost criteria in [10] and proposed two modified cost functions - *sum of distances* and *leader distance*. A followup paper by Kargar et al [9] further extended the framework of [8] and included the cost of using an expert's services in conjunction with communication cost. The work by Li et al [11] extended the work of Kargar et al [10] in the direction of generalized task where a specific number of experts are required against each skill. A similar extension was studied by Gajewar et al [6] using a slightly different communication cost function which is based on the graph density and is a generalization of the cost function used in [10]. Recently, Anagnostopoulos et al [1] extended the work of Lappas et al [10] where tasks arrive in an online fashion and it is required to balance the work load across team members. In a related paper, Majumder et al [12] defined a notion of capacity for each expert and put an additional constraint of capacity violation.

To summarize, the initial work of Lappas et al [10] and all its subsequent extensions [8, 9, 11, 6, 1, 12] formulated the team formation problem as some kind of a combinatorial optimization problem and gave approximation algorithms for the same, as the original problems turned out to be NP-hard. For all these formulations, *skill cover* is a default constraint which ensures that a required set of skills are available in the selected team. Further, all these formulations maximize some kind of a *quality* score for the selected team which encapsulates two key dimensions - social compatibility and the formation cost of the team which is measured via some kind of a communication cost amongst team members and via the included experts' service costs respectively.

1.2 Contributions and Outline In this paper, we propose a generic formulation based on the theory of submodular function optimization [5] [1]. Our motivation stems from that fact that the selected team is a subset of the overall pool of experts and hence a natural way to formulate this problem is in the form of optimizing a set function applied to subsets of experts. Our contributions are detailed below.

[1] A brief introduction to submodular functions and related results are given in Appendix A

1. To the best of our knowledge, we show for the first time that most of the important design criteria for this problem- including skill cover, social compatibility, and economy- can be fully modeled within one single formulation as an unconstrained submodular function maximization problem - which is an NP-hard problem for general submodular functions. In particular, we show that the skill cover constraint proposed in [10, 8, 9] can be expressed in the form of a submodular function in our formulation. Moreover, this formulation allows for a softer form of skill constraint - that is, a designer can specify *compulsory* and *optional* skills requirements separately.

2. We show that while its not possible to express the communication cost functions proposed by [10] as submodular or supermodular functions, it is possible to do so for the improved functions suggested by [8, 9], namely *sum of distances* and *leader distance*. We further propose modifications to these two communication cost functions and rectify the limitation that there should be only one designated expert in the team for every skill.

3. We also suggest a few additional naturally arising constraints for the team formation problem and show that they can be folded into the overall submodular formulation as well.

4. Finally, we show that in our formulation, the submodular function is non-negative and non-monotone. Maximization of this class of submodular functions has been explored to a lesser extent than that of its monotone counterpart. A few recent works [7] [4] have come up with randomized approximation schemes for this problem with approximation factors of upto 0.41. In this paper, we customize this algorithm to our setting and conduct an extensive set of experiments to show its efficacy.

Our formulation offers several advantages and allows for further extensions over the existing formulations.

1. **Skill Cover Softening:** In our formulation, covering the required skill set is a soft constraint in the sense that we allow designer specifying an importance level for each skill to control the trade-off between skill coverage and communication cost or economy. A skill is dropped in the final composition of the team if there are significant logistical benefits from doing so like having an especially smaller team size or much better connectivity within the team. The designer can enforce a skill requirement as a hard constraint by setting up very high importance levels.

2. **Better Team Communication:** Lappas et al [10] assumed that social connectivity between any two team members should only include the connections they share via other team members. Like [8], our formulation allows communication among team members via all the nodes in the entire social graph irrespective of whether they are part of the team or not. This makes sense because the communication cost between two people in the team has nothing to do with the team itself. Two people can still be compatible if they have low communication cost in the overall social network even if their mutual connections are not a part of the same team.

3. **Connectivity Relaxation:** Unlike many existing formulations, we never enforce the requirement that the selected team should represent a connected subgraph of the given social network. In normal circumstances, this connectivity is likely to happen anyway but for certain cases, this requirement may introduce unnecessary redundancy in the team. For instance, if two people with a mutual contact are enough for the team, there is no reason for the mutual contact to be in the team as well if he adds no value. This seems like a strong relaxation, but again, as long as the communication cost between any two members is minimized, the path between them shouldn't matter.

2 Problem Setup

Let $V = \{v_1, v_2, \ldots, v_n\}$ be the set of experts in an organization. We use $[V]^k = \{A : A \subseteq V, |A| = k\}$ to denote k-sized subsets of V. Let $S = \{s_1, s_2, \ldots, s_m\}$ be a set of m different skills. Each expert $v \in V$ has a set of skills, denoted by $S_v \subseteq S$. That is, $S_v = \{s \mid s \in S;$ expert v has skill $s\}$. For example, in a software development organization, the people may have skill sets such as {*Python, Ruby, R*}, {*Java, C++*}, {*Haskell, OCaml, Lambda*}, etc.

These experts are connected together in a social network which is modeled as an undirected weighted graph $G = (V, E, W)$. The edge weight $w_{ij} \geq 0$ for an edge between experts v_i and v_j denotes the cost of communication and collaboration between these two experts. The lower the value of w_{ij}, the easier it is for these two individuals to communicate and collaborate. Without loss of generality, we assume that the graph G is connected - we can transform every disconnected subgraph to a connected one by simply adding very high-weight (say, a weight higher than the sum of all pairwise shorted paths in G) edges between every pair of nodes that belong to different connected components.

In what follows, we describe each of the following design requirements and demonstrate a way to express them via submodular/supermodular functions - (i) skill coverage, (ii) social compatibility, (iii) teaming cost, (iv) other miscellaneous requirements.

2.1 Submodularity in Skill Coverage

A project $P \subseteq S$ is defined as a set of skills required to complete the project. A team $T \subseteq V$ is a subset of experts. The team T is said to *cover* the project P if for every skill s required in project P, there is someone in the team T who possesses that skill. Formally, let $S_T = \bigcup_{v \in T} S_v$ be the set of all skills possessed by at least one member in a team T. The team T is said to cover project P, if $P \subseteq S_T$. For each skill $s \in S$, we define a set $V_s \subseteq V$ representing the set of all those experts who posses this skill. For a given skill s, we define an indicator set function $f_{skill}(\cdot, s) : 2^V \mapsto \{0, 1\}$ over the space of all possible teams (i.e. subsets) of the given set of experts. Formally,

$$f_{skill}(T, s) = \mathbf{1}_{\{T \cap V_s \neq \emptyset\}}; \forall s \in S, T \in 2^V \tag{2.1}$$

where $\mathbf{1}_{\{\cdot\}}$ is an indicator function and 2^V denotes the power set of V. This function takes a value 0 for any subset of experts who do not posses a given target skill s and 1 otherwise. An interesting and useful property about this set function is captured in the following lemma.

LEMMA 2.1. *For any given skill $s \in S$, the function $f_{skill}(\cdot, s)$ is a submodular function.*

Proof. This follows from the first order differences based definition of submodular functions given in Appendix A as part of the supplementary material.

A natural extension of the indicator function $f_{skill}(\cdot, \cdot)$ to the skillset required in project P is as follows.

$$f_{skill}(\cdot, P) = \textstyle\prod_{s \in P} f_{skill}(\cdot, s) \tag{2.2}$$

Unfortunately, the function $f_{skill}(\cdot, P)$ is not as well behaved as $f_{skill}(\cdot, s)$ (as shown in the next lemma).

LEMMA 2.2. *The function $f_{skill}(\cdot, P)$ given in Equation (2.2) is neither a submodular nor a supermodular function for any $P \subseteq S$ having $|P| > 1$.*

Proof. See Appendix B.

Note that it is the multiplicative form of this function $f_{skill}(\cdot, P)$, given in Equation (2.2), which takes away its nice properties. Therefore, we are motivated to redefine this function in the additive form as follows. We call this function the *skill coverage* function.

DEFINITION 1. (SKILL COVERAGE FUNCTION) *For a given project $P \subseteq S$, the skill coverage function is defined as follows.*

$$f_{skill}(\cdot, P) = \sum_{s \in P} C_s f_{skill}(\cdot, s) \tag{2.3}$$

where for each skill $s \in P$, the weight $C_s \geq 0$ denotes its importance level.

LEMMA 2.3. *The function $f_{skill}(\cdot, P)$ defined by Equation (2.3) is a submodular function for any $P \subseteq S$.*

Proof. This follows from the fact that a non-negative weighted sum of submodular functions is also a submodular function [5].

In view of the above definition, the skill coverage design requirement of the team formation problem can be expressed in the equivalent form of maximizing the submodular function $f_{skill}(\cdot, P)$ with an appropriate choice of weights C_s. By choosing a higher weight C_s for the skill $s \in P$, we can make this skill a compulsory skill whereas, by choosing a lower weight, we can make it an optional skill. Thus, this formulation avoids the need of having a constraint for the skill coverage and instead it allows folding this constraint into the objective function itself. Moreover, it also allows making the possession of some skills hard constraints and the possession of some others soft constraints. If all the skills in a given project are equally important then we can set $C_s = C \; \forall s \in P$. Choosing a sufficiently high value for C would ensure picking all the required skills in the team selected.

2.2 Submodularity in Social Compatibility Social compatibility is the second most important design consideration in the team formation problem. Social compatibility can be expressed in the form of a communication cost as suggested by many recent works [10, 8, 9, 11, 6, 1]. In what follows, we list important cost functions that have either been proposed in the literature or are quite natural in the context.

- **Diameter [10]:** Given a team $T \subseteq V$, the diameter of the team is defined as

  $$Diameter = \max_{(v_i, v_j) \in T \times T} d_T(v_i, v_j) \tag{2.4}$$

  where $d_T(v_i, v_j)$ is the shortest path distance between vertices v_i and v_j over the subgraph G_T of the given social graph obtained by restricting on T.

- **Minimum Spanning Tree (MST) [10]:** Given a team $T \subseteq V$, the MST cost is the cost of the *minimum spanning tree* on the subgraph G_T.

Its important to note that both of these costs depend on the subgraph G_T induced by the selected team T. Additionally, these two cost functions have some limitations as pointed out by Kargar et al [8]. Motivated by this, Kargar et al [8] suggested two modified cost functions - *sum of distances* and *leader distance*. These modified cost functions are based on a critical assumption - for every skill s that is required in project P, there is only one expert v_s in the team T, who is designated to cover this skill in this team. It is possible that the same expert may cover multiple skills.

- **Sum of Distances [8]:** Given a team $T \subseteq V$, the sum of distances for this team is defined as

  $$sumDistance = \sum_{(s_i, s_j) \in P \times P} d(v_{s_i}, v_{s_j})$$

  where $d(v_{s_i}, v_{s_j})$ denotes the shortest possible length of a path in the given social graph which connects two team members such that one of them is designated to cover the skill s_i and the other one is designated to cover the skill s_j.

- **Leader Distance [8]:** Given a project $P \subseteq S$, a team $T \subseteq V$, and a leader $l \in V$, the leader distance of T with respect to l is defined as

  $$leaderDistance = \sum_{s \in P} d(v_s, l)$$

  where $d(v_s, l)$ denotes the length of the shortest path (in the given social graph) between leader l and the designated team member for skill s.

Note that for both of these cost functions, the distance between two experts is defined over the whole graph G and not over the subgraph G_T. Also, for a given team T comprising of expert(s) covering each and every skill required in project P, the above costs depend not just on T and P alone but also the mapping of skills to the team members who cover these skills. We further extend these two cost functions in order to relax the assumption that every skill is covered by only one expert in the team and this mapping between the skills and the team members is announced a priori.

- **Modified Sum of Distances:** For a team $T \subseteq V$, the modified sum of distances is given as

  $$modSumDistance = \sum_{(s_i, s_j) \in P \times P} d(s_i, s_j, T)$$

  where $d(s_i, s_j, T)$ is the length of the shortest path (in the given social graph) between two team members such that one of them has skill s_i and the other one has skill s_j. More formally,

  $$d(s_i, s_j, T) = \min_{\substack{(v_i, v_j) \in T \times T \\ v_i \in V_{s_i}, v_j \in V_{s_j}}} d(v_i, v_j)$$

  where $d(v_i, v_j)$ is the shortest path length between vertices v_i and v_j in the given social graph. Note

that this function considers the communication cost between pairs of skills rather than between pairs of team members. Often, the communication between certain pairs of skills are more critical than others and hence require extra care to minimize the distance between such critical pairs of skills. For example, in a project, there could be two sub-groups of the team members, one handling the theoretical part and the other handling the implementation part. In such a scenario, the members of any sub-group need to communicate among themselves more often than the members of the other sub-group. This can be achieved by extending the above function so as to imbue each pair of skills with a positive weight which depends on how critical it is for the members responsible for those skills to work together. Thus, we can define a weighted version of the Modified Sum of Distances as $\sum_{(s_i,s_j)\in P\times P} w_{ij}\ d(s_i, s_j, T)$, where $w_{ij} \geq 0$ is the weight for skill pair (s_i, s_j).

- **Modified Leader Distance:** Given a project $P \subseteq S$, a team $T \subseteq V$, and a leader $l \in V$, the modified leader distance is given by

$$modLeaderDistance = \sum_{s\in P} d(s, l, T)$$

where $d(s, l, T)$ denotes the shortest possible length of a path in the given social graph which connects leader l with a team member who has skill s. More formally, $d(s, l, T) = \min_{v\in T\cap V_s} d(v, l)$. Similar to the modified sum of distances function, we also define the weighted version of the modified leader distance as $\sum_{s\in P} w_s\ d(s, l, T)$, where $w_s \geq 0$ is the weight assigned to the particular skill s.

- **Sum of Degrees:** This is a new cost function introduced by us. For a given team T, this cost is given by the negative of the sum of the degrees of all the team members. For any given team member, while counting the degree, we include all his neighbors in the entire social graph irrespective of whether they are a part of the given team T or not. It is motivated by the fact that the more connected a member is, the more likely he is to be a good team player and therefore we aim towards minimizing this function.

Given a project P, and a social graph G, let the function $f_{social}(\cdot, P, G) : 2^V \mapsto \mathbb{R}^+$ denote the communication cost function of a team. We call this function the *social compatibility cost function* which can be expressed as any of the above five cost functions. Ensuring social compatibility in the team implies minimizing this social compatibility cost function. In what follows, we highlight the submodular/supermodular properties of this function.

LEMMA 2.4. *Given a project P and a social graph G, the social compatibility cost function $f_{social}(\cdot, P, G)$ is*

1. *Neither submodular nor supermodular if it is defined as either the diameter or the MST cost.*
2. *Modular if it is defined as the sum of degrees.*
3. *Supermodular if it is defined as the sum of distances and modular if it is defined as the leader distance with a pre-defined leader.*
4. *Supermodular if it is defined as either the modified sum of distances or the modified leader distance.*

Proof. See Appendix B.

Note, conic combinations preserve submodularity. Therefore, any non-negative weighted combination of the modular or supermodular functions defined above would also be a supermodular social compatibility cost function. Also note that the negative of a supermodular function is a submodular function, and minimizing the above supermodular functions is equivalent to maximizing the negative of these functions.

2.2.1 A Note on the Choice of Cost Function From the previous lemma, it appears as though the diameter and the MST cost functions cannot be captured within the submodular functions framework. However, we would like to mention that these two functions have certain limitations, such as instability, and these limitations were rectified in the improved functions proposed by Kargar et al [8] like the sum of distances and the leader distance functions, which, on the other hand, are supermodular. Thus, we can say that while some of the original cost functions to measure social compatibility cannot be modelled in the proposed submodular framework, their modified counterparts can be. The submodular framework, therefore, still holds ground and is useful despite a negative result in Lemma 2.4.

2.3 Submodularity in Teaming Cost The next important consideration in the team formation problem is *teaming cost* which can be measured in many ways.

- **Team Size:** Assuming each expert is as valuable (or expensive) as any other expert, a simple way to measure teaming cost would be its size.
- **Personnel Cost [9]:** For a given team T, and a given project P, the cost of availing the services of experts for this project can be modeled as $\sum_{v\in T}\sum_{s\in P\cap S_v} cost(v, s)$, where $cost(v, s)$ is the cost of availing the services of expert v for skill s.

If we let $f_{team}(\cdot, P) : 2^V \mapsto \mathbb{R}^+$ denote the cost of forming team for a given project by using any of the above functions, our goal should be to minimize the function $f_{team}(\cdot, P)$. In what follows, we highlight the submodular properties of the teaming cost function.

LEMMA 2.5. *The function $f_{team}(\cdot, P)$ is a modular function if it is defined as either team size or the personnel cost.*

Proof. See Appendix B.

2.4 Submodularity of Miscellaneous Requirements Apart from skill coverage, social compatibility, and teaming cost, there could be many other design considerations while forming a team. In what follows, we highlight two such obvious considerations which, to the best of our knowledge, have not been looked in the literature so far. We also show that these considerations can be captured within the submodularity framework.

- **Redundant Skills Avoidance:** Recall that the skill coverage just requires having an expert in the team for every skill required in a project. This does not rule out the possibility that an expert may have multiple skills in the selected team. It is quite possible in some situations that a project requires heterogeneity in the pool of skills available in the selected team. For example, this can help balancing the work load across the team members [1]. For such a case, we can define a cost function called *team redundancy cost* $f_{red}(\cdot) : 2^V \mapsto \mathbb{R}^+$ as follows

$$f_{red}(T) = \textstyle\sum_{(v_i,v_j)\in T\times T} r(v_i, v_j) \tag{2.5}$$

  where $r(v_i, v_j)$ measures the pairwise overlap of skills between team members v_i and v_j. In the team formation problem, our goal should be to minimize this function.

- **Inclusion of Selected Experts:** In many situations, a project may require the inclusion of a predefined set of experts in the final team for various reasons. This requirement can be captured by means of defining a function called *experts inclusion* $f_{expInc}(\cdot) : 2^V \mapsto \mathbb{R}^+$ as follows

$$f_{expInc}(T) = \textstyle\sum_{v\in T} w_v \tag{2.6}$$

  where $w_v \geq 0$ is a weight assigned to an expert $v \in V$. We can always assign very high positive weights to those experts whose inclusion is a must in the final team and set a very low weight for all the other experts. In the team formation problem, our goal should be to maximize this function.

LEMMA 2.6. *$f_{red}(\cdot)$ is a supermodular function and $f_{expInc}(\cdot)$ is a modular functions.*

Proof. See Appendix B.

2.5 The Overall Formulation Based on the submodular structure shown so far for various design considerations, it is apparent that the team formation problem can be posed as the following optimization problem

$$\max_{T\in 2^V} f_{overall}(T, P, G) \tag{2.7}$$

where, $f_{overall}(\cdot, P, G) : 2^V \mapsto \mathbb{R}$ is a submodular function encapsulating all the design considerations. That is,

$$\begin{aligned} f_{overall}(\cdot, P, G) &= \alpha_{skill} f_{skill}(\cdot) - \alpha_{social} f_{social}(\cdot, P, G) \\ &- \alpha_{team} f_{team}(\cdot, P) - \alpha_{red} f_{red}(\cdot) \\ &+ \alpha_{expInc} f_{expInc}(\cdot) \end{aligned}$$

Non-negative coefficients α_{skill}, α_{social}, α_{team}, α_{red}, $\alpha_{expInc} \geq 0$ represent the relative importance for these design considerations. By making use of previous lemmas, it is easy to verify that $f_{overall}(\cdot, P, G)$ is a submodular function. Formulation (2.7) suggests that the team formation problem, in general, can be posed as an unconstrained submodular function maximization problem which is an NP-hard problem for arbitrary submodular functions [5]. A few critical observations are in order with regard to this formulation.

2.5.1 Non-Negative Non-Monotone Function It is easy to see that the function $f_{overall}(\cdot, P, G)$ could be non-monotone in general. For example, inclusion of an extra member to the set can increase the value of the function if this new member adds a hitherto absent skill and does not increase the social cost too much. However, the reverse of this is also possible if the new member brings no new skill to the team but adds a high social cost for its inclusion. Further, this function $f_{overall}(\cdot, P, G)$ could be negative as well. However, it is also easy to check that for a given project P and a graph G, this function is bounded below by a constant on the negative side because each of its component function is bounded below by a constant. Using this lower bound on this function, we can transform this function by adding a constant so that it always stays positive. Note that adding a constant does not change the property of submodularity. From now onwards, we will adhere to the convention that *$f_{overall}(\cdot, P, G)$ is non-negative and non-monotone.*

2.5.2 Skill Coverage as a Hard Constraint In this formulation, skill coverage is a soft constraint. This

implies that some of the skills required for a project P could be missing in the optimal solution. This can, however, be easily overcome simply by choosing sufficiently high values for the coefficients α_{skill}, and C_s in the overall formulation which would effectively make the skill coverage constraint a strict constraint.

2.5.3 Generalized Task This formulation can also be used for the scenario where it is required to have a certain minimum number of experts against each skill. For this, we can augment the definition of the project P where we replicate a skill, say s, as many times as the required minimum number of experts for that skill. If minimum number of experts required for this skill is n_s then we assign a different and unique label to each of these n_s replicas of the skill s, denote them by $s^1, s^2, \ldots, s^{n_s}$. Similarly, for all the experts who possess the skill s, we replicate them into n_s different users with identical properties except that the i^{th} replica possesses the skill s^i instead of skill s.

In view of this extension, there is no loss of generality even if we consider the team formation problem where it is required to have at least one expert in the team for every skill in project P. All approximation algorithms presented in the next section are designed for this setting.

3 Algorithm

In general, submodular function maximization is an NP-hard problem. For monotone submodular functions, unconstrained maximization is trivial and approximations for constrained maximization have been well explored in the literature. For example, the well known greedy scheme of Nemhauser et al [13] renders an approximation factor of $(1 - 1/e)$. However, the problem of finding the maximum for an arbitrary non-monotone submodular function with or without constraints is notoriously hard even to approximate. To the best of our knowledge, approximation schemes for unconstrained maximization of non-monotone submodular function have only been obtained for functions which are non-negative [4, 7]. A 2/5-approximation algorithm for such problems were first given by Feige et al [4]. This is a randomized local search algorithm. Gharan and Vondrák [7] introduced a similar algorithm based on simulated annealing which improved the approximation factor to 0.41. We have adopted Gharan and Vondrák's algorithm to get an approximate solution for the general team formation problem given in (2.7). A high level description of this algorithm is given in Algorithm 1. In what follows, we describe the flow of this algorithm. For brevity, we denote $f_{overall}$ simply as f. The algorithm starts with a given set of experts V, a social graph G, and the objective function

Algorithm 1 Simulated Annealing Algorithm for Team Formation Problem

Input: Set of experts V, social graph G, project P, function $f(\cdot, P, G) : 2^V \to \mathbb{R}^+$
Output: Team T^*

1: $T^* \leftarrow \emptyset$
2: $T \leftarrow \emptyset$
3: **for** $p \leftarrow 1/2;\ p \leq 1;\ p \leftarrow p + \theta$ **do**
4: **while** $\exists v \in V$ such that $\widehat{E}_p[f(R_p(T\Delta\{v\}))] \geq \widehat{E}_p[f(R_p(T))]$ **do**
5: $T \leftarrow T\Delta\{v\}$
6: **if** $f(T^*) \leq f(T)$ **then**
7: $T^* \leftarrow T$
8: **end if**
9: **if** $f(T^*) \leq f(V - T)$ **then**
10: $T^* \leftarrow V - T$
11: **end if**
12: **end while**
13: **end for**
14: **return** T^*

$f_{overall}(\cdot, P, G) : 2^V \to \mathbb{R}^+$. Note that we have assumed that the function $f_{overall}(\cdot, P, G)$ has already been converted into a non-negative function by adding a sufficiently high constant as discussed earlier.

The algorithm begins with an empty team T^* and a continuously changing placeholder team T. The **for** loop (Steps 2-13) iterates over the probability parameter p starting from $1/2$ and going all the way upto 1 with an increment of θ in every step. Typically, we assume the quantization parameter $\theta = 1/n^k$ where $n = |V|$ and k is typically chosen as 3.

For each value of p, given a seeding set T, define $R_p(T)$ as the set obtained by deleting elements from T and adding elements to T independently with probability $(1 - p)$. That is, $R_p(T)$ is a set obtained from T such that $\forall v \in T, v \in R_p(T)$ independently with probability p and $\forall v \notin T, v \in R_p(T)$ independently with probability $(1 - p)$. During Steps 4-12, we continuously look for an element $v \in V$ such that its addition to the set T (if it is not already present in the set T) or deletion from the set T (if it is present in T) would increase the expected value of the function $f(R_p(T, P, G))$. That is, we look for an element v such that $E_p[f(R_p(T\Delta\{v\}))] \geq E_p[f(R_p(T))]$, where Δ denotes the symmetric difference of two sets such that $T_1\Delta T_2 = (T_1 \backslash T_2) \cup (T_2 \backslash T_1)$. In practice, we work with an estimate of $E_p[f(R_p(\cdot))]$, denoted by $\widehat{E}_p[f(R_p(\cdot))]$, which is obtained via sampling.

For every element $v \in V$ and the set $T \subseteq V$ such that condition of **while** loop (Step 4) is true, we check whether the objective function value can be increased

by replacing T^* with $T\Delta\{v\}$ or $V \setminus (T\Delta\{v\})$ during the Steps 5-10. If yes, then we replace T^* with the appropriate set.

The Algorithm 1 has been shown to return a 0.41-approximate solution with high probability [7].

3.1 Insights About the Algorithm In this section, we bring out some finer points about this algorithm.

Note that in practice, searching for an element in Step 4 of the algorithm can be done in a systematic manner as it is order independent. In our case, we first search over all the experts outside the current set T and consider them for inclusion and then subsequently, we consider all the experts within the current set T and consider them for exclusion. However, the condition in the while loop might cause the algorithm to cycle forever. To prevent this, we can also impose a bound $\kappa_{\max}$ on how many times the search is conducted before changing the value of the parameter p.

At the end, when we have $p = 1$. It is easy to see that $R_{p=1}(T) = T$ and therefore $\widehat{E}_p[f(R_{p=1}(T))] = f(T)$. In this special case, this algorithm becomes the deterministic local search algorithm presented by Feige et al [4] whose approximation factor is at least $1/3$.

Another point to note is that we never really need the exact value of the constant which is added to the function $f_{overall}(\cdot, P, G)$ to make it non-negative at any step of the algorithm. The only times when the actual function value is computed is either in Step 4, 6, or 9. In all these cases, it can be seen that the constant term appears on both sides of the inequality and hence gets cancelled out. Therefore, we never really need to specify the exact value of this constant.

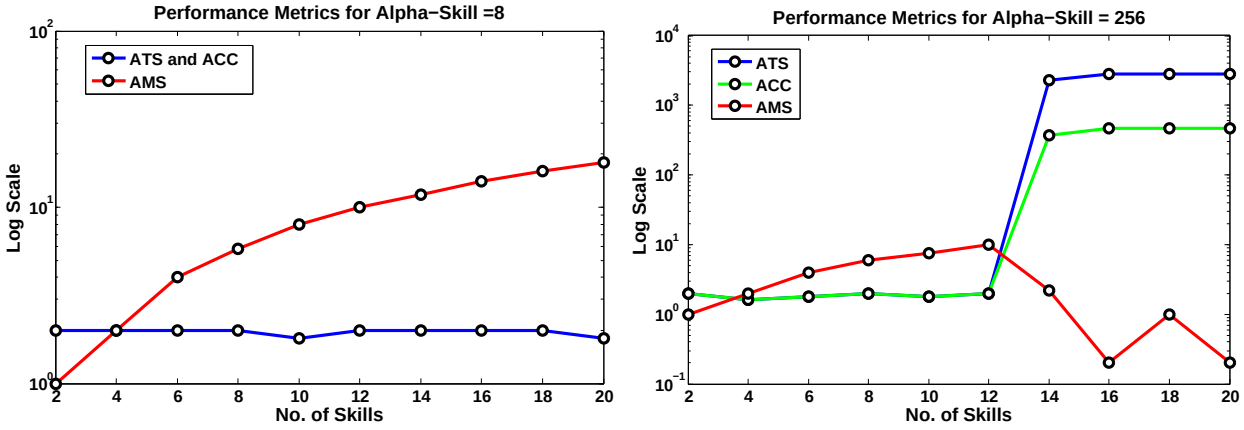
Finally, the last point is regarding the skill weights C_s. Since, we are performing a repeated search over the entire set V, if C_s are chosen high enough, the algorithm will end up including experts in the team until all the skills get covered. In fact, for those skills which are hard constraints, we might as well modify the T^* updating conditions in Algorithm 1 to include the constraint that the T^* will be updated to T (or $V \setminus T$) only if T (or $V \setminus T$) covers more skills than T^*.

4 Experiments

We used a snapshot of DBLP data taken on July 31, 2013 to conduct our experiments. Our experimental setup is pretty much similar to [10]. We worked with only those entries in DBLP dataset which correspond to papers published in the areas of *Database (DB), Data Mining (DM), Artificial Intelligence (AI), and Theory (T)* conferences. For each of these four areas, we considered the following conference venues: DB={SIGMOD, VLDB, ICDE, ICDT, EDBT}, DM={WWW, KDD, SDM, PKDD, ICDM}, AI={ICML, ECML, COLT, UAI}, and T={SODA, FOCS, STOC, STACS}. We refer to these selected sets of papers as the **DBLP** dataset. The input for the team formation problem is generated as follows. The set of experts V consists of the set of authors that have *at least 3 papers* in the **DBLP** dataset. The skillset S_v of an expert $v \in V$ consists of the set of terms that appear in *at least two titles* of papers in **DBLP** dataset that he has co-authored. This procedure creates a set V consisting of 9186 individuals and 4013 distinct skills. We created a social graph $G = (V, E)$ over this set V of experts where two experts v_i and v_j are connected by an edge if they have co-authored at least 2 papers. This resulted in 19642 edges over these 9186 nodes. The weight w_{ij} of an edge connecting vertices v_i and v_j is given by $1 - |z_i \cap z_j|/|z_i \cup z_j|$, where z_i and z_j are the total number of papers authored by v_i and v_j, respectively. The edge weights essentially represent the Jaccard distance between experts and the shortest path distance between two experts is computed using these pairwise Jaccard distances.

4.1 Performance Evaluation In this section, we evaluate the performance of the simulated annealing algorithm for the team formation problem given in Algorithm 1. We evaluate its performance on three metrics - *team size, missing skills, and connectivity.*

In our experiments, we generated a project P as follows. We chose one of the research areas among DB, DM, AI, and T. Then, we randomly picked t required skills from the terms appearing in papers published in conferences belonging to this area. We report the results for $t = \{2, 4, \ldots, 20\}$. For each t, we generated 100 random projects and computed the *Average Team Size (ATS)*, *Average Number of Missing Skills (AMS)*, and *Average Number of Connected Components (ACC)* by using the simulated annealing based algorithm. These results are reported in Figure 1. By missing skills, we mean the skills in the given project P that are not present in any of the team members. In our implementation, we chose $\theta = 0.1$. We repeated these experiments with varying value of α_{skill} but fixed value of $\alpha_{team} = \alpha_{social} = 1$. Observe in Figure 1 that as we increase α_{skill}, the ATS starts increasing because we are putting more emphasis on meeting required skills. As a consequence, the AMS value starts decreasing. As far as ACC is concerned, it stays below ATS for $\alpha_{skill} = 256$ but coincides with ATS for $\alpha_{skill} = 8$. Therefore, we can say that as α_{skill} increases, the team starts becoming more and more connected. These experiments suggest an important point that choosing the right values of these hyper-parameters C_s, α_{social}, and α_{team} is quite crucial and it could be somewhat messier. An interesting open problem at this moment



(a) ATS, AMS, and ACC Scores for $\alpha_{skill} = 8$

(b) ATS, AMS, and ACC Scores for $\alpha_{skill} = 256$

Figure 1: Performance of Simulated Annealing Based Algorithm for Team Formation Problem

would be to automatically learn the right values for these hyper-parameters.

5 Conclusions and Future Work

This papers shows that the well known team formation problem can be formulated as an unconstrained submodular function maximization problem in its full generality, where the objective function is non-negative and non-monotone. The merit of this formulation lies in the fact that it provides lots of modeling flexibility and robustness. A wide range of design criteria can be added or removed from the model without altering the algorithm. This work opens up plenty of opportunities for further investigations. In particular, setting the relative weights of various design considerations is a daunting task in practice. Moreover, the proposed simulated annealing algorithm's running time could be unacceptable in certain worst-case situations and it still remains an open question whether one can improve on this without compromising much with solution quality.

Acknowledgments

Authors would like to acknowledge Prof. H. Narayanan, IIT Bombay for several useful discussions.

References

[1] A. Anagnostopoulos, L. Becchetti, C. Castillo, A. Gionis, and S. Leonardi. Online team formation in social networks. In *WWW 2012*, Lyon, France, April 2012.

[2] A. Baykasoglu, T. Dereli, and S. Das. Project team selection using fuzzy optimization approach. *Cybernetics and Systems*, 38(2):155 –185, 2007.

[3] S. J. Chen and L. 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.

[4] U. Feige, V. S. Mirrokni, and J. Vondrak. Maximizing non-monotone submodular functions. *SIAM Journal on Computing*, 40(4):1133–1153, 2011.

[5] S. Fujishige. *Submodular Functions and Optimization*, volume 58 of *Annals of Discrete Mathematics*. Elsevier, second edition, 2005.

[6] A. Gajewar and A. D. Sarma. Multi-skill collaborative teams based on densest subgraphs. In *SDM*, 2012.

[7] S. O. Gharan and J. Vondrák. Submodular maximization by simulated annealing. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1098–1116. SIAM, 2011.

[8] M. Kargar and A. An. Discovering top-k teams of experts with/without a leader in social networks. In *CIKM 2011*, Glasgow, Scotland, UK, October 2011.

[9] M. Kargar, A. An, and M. Zihayat. Efficient bi-objective team formation in social networks. In *ECML/PKDD 2012*, 2012.

[10] T. Lappas, K. Liu, and E. Terzi. Finding a team of experts in social networks. In *KDD 2009*, July 2009.

[11] C.-T. Li and M.-K. Shan. Team formation for generalized tasks in expertise social networks. In *IEEE International Conference on Social Computing*, 2010.

[12] A. Majumder, S. Datta, and K. Naidu. Capacitated team formation problem on social networks. In *KDD*, pages 1005–1013, 2012.

[13] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions - I. *Mathematical Programming*, 14(1):265–294, 1978.

[14] H. Wi, S. Oh, J. Mun, and M. Jung. A team formation model based on knowledge and collaboration. *Expert Syst. Appl.*, 36(5):9121–9134, 2009.

[15] A. Zzkarian and A. Kusiak. Forming teams: an analytical approach. *IIE Transactions*, 31:85–97, 2004.

Appendix

Please refer to `https://sites.google.com/site/avradeepbhowmik1001/research`.