Discovering Top-k Teams of Experts with/without a Leader in Social Networks

Mehdi Kargar and Aijun An
Department of Computer Science and Engineering
York University, Toronto, Canada
{kargar,aan}@cse.yorku.ca

ABSTRACT

We study the problem of discovering a team of experts from a social network. Given a project whose completion requires a set of skills, our goal is to find a set of experts that together have all of the required skills and also have the minimal communication cost among them. We propose two communication cost functions designed for two types of communication structures. We show that the problem of finding the team of experts that minimizes one of the proposed cost functions is NP-hard. Thus, an approximation algorithm with an approximation ratio of two is designed. We introduce the problem of finding a team of experts with a leader. The leader is responsible for monitoring and coordinating the project, and thus a different communication cost function is used in this problem. To solve this problem, an exact polynomial algorithm is proposed. We show that the total number of teams may be exponential with respect to the number of required skills. Thus, two procedures that produce top-k teams of experts with or without a leader in polynomial delay are proposed. Extensive experiments on real datasets demonstrate the effectiveness and scalability of the proposed methods.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Data Mining; D.2.8 [**Software Engineering**]: Metrics

General Terms

Algorithms, Experimentation

Keywords

Team Formation, Approximation Algorithms, Social Networks

1. INTRODUCTION

A project cannot be successfully completed without effective communications and collaborations among its team members. Given a project whose completion requires a set of skills, we tackle the problem of finding from a social network a team of individuals that not only cover all the required skills but can also communicate and collaborate effectively. The social network is modeled as a graph whose nodes represent experts, each with one or more skills, and whose edge between two nodes is weighted by the communication cost between the two corresponding experts.

This problem has been introduced in the data mining community by Lappas et al. [10]. They defined two functions for estimating the communication cost of a team. The first function uses the diameter of the subgraph formed by the team to measure the team communication cost, which is the largest shortest path between any two nodes in the subgraph. The second function uses the cost of the minimum spanning tree (MST) on the subgraph. However, such functions may not measure the communication cost well, especially when the project requires that the holders of each pair of the required skills communicate with each other to perform the corresponding tasks in the project. The diameter function only measures the communication cost between the two experts that are furthest away from each other, and the MST function does not measure the cost of all the required communication either. Another disadvantage of these functions is their instability: a slight change in the graph may result in a radical change in the solution. At the same time, the diameter and MST based functions may be insensitive to adding or deleting a connection in the graph since they only measure part of the communication cost.

In this paper, we introduce two new cost functions. We consider two types of communication structures within a team. Assume that each required skill corresponds to a task in the project. In the first communication structure, the experts for each pair of required skills need to communicate to each other to complete the corresponding tasks. For such a structure, we define a cost function, called *Sum of Distances*, to measure the communication cost of a team using the sum of the shortest distances between the experts for each pair of skills. In the second type of communication structure, a leader needs to communicate with each team member to monitor and coordinate the project. For such a structure we define a cost function, named *Leader Distance*, that computes the sum of shortest distances between the leader and each skill holder in the team.

Based on these two new cost functions, we define two problems of team formation from a social network. One is to find a team of experts without a leader which covers the required skills and minimizes the *Sum of Distances* function.

CIKM'11, October 24–28, 2011, Glasgow, Scotland, UK.

The other is to find a team of experts with a leader which covers the required skills but minimizes the *Leader Distance* function. We propose algorithms for solving both problems. In many situations, the user is not satisfied with only one answer. Instead, he/she is interested in exploring top-k answers. Previous work in this area only found one single best answer [10]. We present two procedures that enumerate top-k teams of experts with or without a leader in polynomial delay. The contributions of this paper are summarized below:

1. We propose two new functions for measuring the communication cost of a team of experts in social networks. The *Sum of Distances* function considers communication costs between each pair of skill holders. The *Leader Distance* function considers the costs between a leader and each of the skill holders in the team.

2. We prove that the problem of finding a team of experts that minimizes the *Sum of Distances* function is NP-hard, and propose an approximation algorithm with approximation ratio of two to solve the problem.

3. We introduce the problem of finding a team of experts with a leader that minimizes the *Leader Distance* function. An exact polynomial algorithm for finding the best team with a leader is proposed.

4. To generate more answers, we design two procedures that enumerate top-k teams of experts with or without a leader in polynomial delay.

5. We conduct extensive experimental evaluations on real data sets that show the effectiveness of our methods.

The paper is organized as follows. Problem statements and definitions are given in section 2. The algorithms for finding the best team of experts with/without a leader are presented in sections 3 and 4. Procedures for enumerating top-k teams of experts are presented in section 5. The experimental results are illustrated in section 6. Related work is presented in section 7 and section 8 concludes the paper.

2. PROBLEM STATEMENTS

Let $C = \{c_1, c_2, \ldots, c_n\}$ denote a set of n experts, and $S = \{s_1, s_2, \ldots, s_m\}$ denote a set of m skills. Each expert c_i has a set of skills, denoted as $Q(c_i)$, and $Q(c_i) \subseteq S$. If $s_j \in Q(c_i)$, expert c_i has skill s_j. In addition, a subset of experts $C' \subseteq C$ have skill s_j if at least one of them has s_j. For each skill s_j, the set of all experts having skill s_j is denoted as $C(s_j) = \{c_i | s_j \in Q(c_i)\}$. A project $P \subseteq S$ is defined as a set of skills required to complete the project. A subset of experts $C' \subseteq C$ is said to *cover* a project P if $\forall s_j \in P \; \exists \; c_i \in C', s_j \in Q(c_i)$.

The experts in C are connected together in a social network, which is modeled as an undirected and weighted graph G. Each node in G represents an expert in C. In the following, terms *expert* and *node* are used interchangeably. Two nodes in G are connected by an edge if the experts have collaborated or communicated before. The weight of an edge represents the communication cost between two experts. The lower the weight of the edge between two nodes, the more easily the two experts can collaborate or communicate, and the lower the communication cost between them.

The **distance** between two nodes c_i and c_j, denoted as $d(c_i, c_j)$, is the sum of the weights on the shortest path between them in G. It should be noted that the shortest distance function is a metric and satisfies the triangle inequality. If c_i and c_j are not connected in G (directly or indirectly), the distance between them is set to a large value, bigger than the sum of all the pairwise shortest distances between two connected nodes in G [10]. In addition, the **distance** between a node c_i and a subset of nodes C' is defined as $d(c_i, C') = \min d(c_i, c_j) \; \forall \; c_j \in C'$. The node that has the closest distance to c_i among the nodes in C' is defined as $N(c_i, C') = c_x \mid \forall \; c_j \in C' \; d(c_i, c_x) \leq d(c_i, c_j)$. If $C' = \emptyset$, we define $d(c_i, C') = \infty$, and $N(c_i, C') = \emptyset$. Now, we are ready to formally define a team of experts and the problems we tackle.

2.1 Team Formation without a Leader

DEFINITION 1. ***(Team of Experts)*** *Given a set of experts C and a project P that requires a set of skills $\{s_1, s_2, \ldots, s_p\}$, a* team of experts *for P is a set of p skill-expert pairs: $\{\langle s_1, c_{s_1}\rangle, \langle s_2, c_{s_2}\rangle, \ldots, \langle s_p, c_{s_p}\rangle\}$, where c_{s_j} is an expert in C having skill s_j for $j = 1, \ldots, p$. A skill-expert pair $\langle s_i, c_{s_i}\rangle$ means that expert c_{s_i} is responsible for skill s_i in the project.*

According to this definition, for each skill s_j in project P, there is one and only one skill-expert pair in the team that contains s_j. In other words, each skill is coupled with exactly one expert, but an expert may be assigned to more than one skill in the project. Clearly, the number of possible teams is $O(|C_{max}|^p)$, where $|C_{max}| = \max |C(s_i)| \; 1 \leq i \leq p$ and $|C(s_i)|$ denotes the cardinality of set $C(s_i)$. This value is exponential with respect to the number of required skills, i.e. p. Thus, it is not feasible to produce all the teams and rank them according to the cost function.

DEFINITION 2. ***(Sum of Distances)*** *Given a team T of experts for a project: $\{\langle s_1, c_{s_1}\rangle, \langle s_2, c_{s_2}\rangle, \ldots, \langle s_p, c_{s_p}\rangle\}$, the* sum of distances *of T is defined as*

$$sumDistance = \sum_{i=1}^{p} \sum_{j=i+1}^{p} d(c_{s_i}, c_{s_j})$$

In other words, the sum of distances of a team of experts is the sum of the shortest distances between the experts responsible for each pair of skills. It measures the communication cost of the team, assuming that the experts with each pair of required skills need to communicate in order to coordinate on the two corresponding tasks in the project.

PROBLEM 1. ***(Team Formation without a Leader)*** *Given a project P and a graph G representing the social network of a set of experts C, the problem of team formation without a leader is to find a team of experts T for P from G so that the communication cost of T, defined as the sum of distances of T, is minimized.*

Below we compare the sum of distances with two other functions defined in [10]. Suppose the two subgraphs G_1 and G_2 in Figure 1 represent two teams of experts for a project that requires skills a, b, c and d. Experts P and T are responsible for skills a and b in the two teams, respectively. The other experts are each responsible for only one

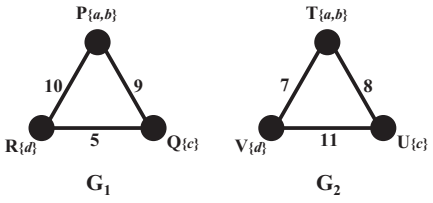

Figure 1: Two teams for project $\{a, b, c, d\}$.

skill (c or d). Suppose the weight on each edge is the shortest distance between the two nodes. The diameter of G_1 is 10, while the diameter of G_2 is 11. The MST cost of G_1 and G_2 is 5+9=14 and 7+8=15 respectively. Thus, the algorithm that minimizes the diameter or MST considers G_1 as the better team. Let d_{i-j} denote the shortest distance between the experts for skills i and j in a team. The *sum of distances* of G_1 is equal to $d_{a-b} + d_{a-c} + d_{a-d} + d_{b-c} + d_{b-d} + d_{c-d} = 0 + 9 + 10 + 9 + 10 + 5 = 43$. Using the same method, the *sum of distances* of G_2 is 41. Thus, with the *sum of distances* function, G_2 is chosen as the better team. As can be seen, the diameter and MST cost functions take into account only part of the communication cost in a team, while the sum of distances adds up the costs between experts for each pair of required skills. Also, our cost function is different from a function that would add up the shortest distances between each pair of experts in the team. Such a function would give G_1 a value of $10 + 9 + 5 = 24$ and G_2 a value of $7 + 8 + 11 = 26$, which results in the selection of G_1. The sum of distances function considers each pair of skills in its computation. We argue that this is more reasonable because for each pair of tasks (i.e., skills) the experts responsible for them need to communicate to coordinate on the tasks. For example, expert T in G_2 needs to communicate with expert V twice, one to coordinate between a and d and the other between b and d. Thus, the contribution of an expert to the total communication cost of the team is proportional to the number of tasks he/she performs in the project.

THEOREM 1. *Solving Problem 1 with the* sum of distances *communication cost function introduced in Definition 2 is an NP-hard problem.*

PROOF. We prove that the decision version of the problem is NP-hard. Thus, as a direct result, Problem 1 is NP-hard too. The decision problem asks whether there exists a team C' which covers a project P with the communication cost **at most** w ($w > 0$), for some constant w. The communication cost of C' is defined in Definition 2.

The problem is obviously in NP. We prove the theorem by a reduction from 3-satisfiability (3-SAT)[1]. First, consider a set of m clauses $D_k = x_k \vee y_k \vee z_k$ ($k = 1, \ldots, m$) and $\{x_k, y_k, z_k\} \subset \{u_1, \overline{u}_1, \ldots, u_n, \overline{u}_n\}$. We define an instance of the above problem as follows. For each pair of variables u_i and $\overline{u}_i$, two experts are created. Thus, we have $2 \times n$ experts. For each pair of variables u_i and $\overline{u}_i$, we create one skill s_i ($i = 1, \ldots, n$). Thus, u_i and $\overline{u}_i$ have skill s_i and the only holders of s_i are u_i and $\overline{u}_i$. In addition, for every clause D_k, we create one skill s_{n+k} ($k = 1, \ldots, m$) such that the holders of skill s_{n+k} consists of the triplet of experts associated with those of x_k, y_k and z_k. Therefore, the number of required skills is $n + m$. We set the distance between each variable and its negation (i.e. u_i and $\overline{u}_i$) to $2 \times w$. The distance between other variables is set to $\frac{w}{\binom{n+m}{2}}$. The distance of each node to itself is set to zero.

A feasible solution to the above problem with *sumDistance* at most w is any set of experts such that from each pair of experts corresponding to u_i and $\overline{u}_i$, exactly one is selected and from each triplet of experts corresponding to x_k, y_k and z_k, one is selected. Thus, if there exists a subset of *sumDistance* at most w, then there exists a satisfying assignment for $D_1 \wedge D_2 \wedge \cdots \wedge D_m$. On the other hand, a satisfying assignment apparently determines a feasible set of experts with *sumDistance* at most w. Therefore, the proof is complete.

□

2.2 Team Formation with a Leader

A project team often has a leader who is responsible for monitoring and coordinating the project. In such a case, each expert in the team needs to communicate with the leader to report the progress and discuss issues related to the project. Thus, the communication cost of the team heavily depends on the distance between the leader and each of the project members. We define the *leader distance* to measure such a communication cost.

DEFINITION 3. ***(Leader Distance)*** *Given a project $P = \{s_1, s_2, \ldots, s_p\}$, a team T of experts selected from graph G for P is $\{\langle s_1, c_{s_1}\rangle, \langle s_2, c_{s_2}\rangle, \ldots, \langle s_p, c_{s_p}\rangle\}$. Assume that team T has a leader L, where L is an expert in graph G, which may or may not belong to $\{c_{s_1}, c_{s_2}, \ldots, c_{s_p}\}$. The* leader distance *of T with leader L is defined as*

$$leaderDistance = \sum_{i=1}^{p} d(c_{s_i}, L)$$

In other words, the *leader distance* of a team of experts is the sum of the shortest distances between its leader and the expert for each required skill. We consider each skill (instead of each expert) when computing the leader distance because for each task an expert performs, communication with the leader is often needed. Thus, the more skills an expert is responsible for in the team, the more important his/her cost of communication with the leader is to the total communication cost of the team.

PROBLEM 2. ***(Team Formation with a Leader)*** *Given a project P and a graph G representing the social network of a set of experts C, the problem of* team formation with a leader *is to find a team of experts T for P and an expert L from C as the leader of the team so that the communication cost, defined as the leader distance of T with leader L, is minimized.*

Note that leader L is chosen from C, meaning that it is not necessary for L to have a skill required by project P. Our priority here is to minimize the leadership communication cost, i.e. the leader distance. A leader that can communicate the most effectively with all the other members of the team

[1]Note that the same approach is used in [2] and [9] for proving the NP-hardness of the *multiple choice cover* problem and a graph keyword search problem, respectively.

is most preferable, no matter whether the leader possesses any of the required skills. The number of possible teams of experts with a leader is $O(|C_{max}|^p \times n)$, where p is the number of required skills in the project, n is the number of experts in G and $|C_{max}| = \max|C(s_i)|$ $1 \leq i \leq p$.

It can be proved that $leaderDistance \geq \frac{sumDistance}{p-1}$, where $sumDistance$ is the sum of distances defined in Definition 2 and p is the number of required skills in the project[2]. Thus, minimizing the leader distance is equivalent to minimizing an upper bound of $sumDistance$. A practical benefit of finding a team with the least leader distance is that polynomial algorithms can be designed to find teams that minimize the leader distance. We propose such an algorithm in Section 4 and demonstrate in Section 6.4 that such an exact algorithm outperforms the approximation algorithm for minimizing the diameter, MST or the sum of distances when using the diameter, MST and the sum of distances as performance measures.

3. FINDING BEST TEAM OF EXPERTS WITHOUT A LEADER

Since Problem 1 is an NP-hard problem, we hereby propose an approximation algorithm that finds the best team with 2-approximation. The pseudo code of this algorithm is presented in Algorithm 1. It takes as input a graph G (i.e., the social network) and a project $P = \{s_1, s_2, \ldots, s_p\}$ representing the required skills. In addition, for each skill s_i, the set of experts in G having s_i (i.e., $C(s_i)$) can be obtained by the algorithm through a pre-built inverted index. The algorithm returns a team, denoted as $bestTeam$, whose communication cost is at most twice that of the optimal team. It also returns the communication cost of $bestTeam$.

The algorithm works as follows. It first initializes $bestTeam$ to $\emptyset$ and its communication cost, $leastSumDistance$, to ∞. Then, for each required skill s_i ($i = 1, \ldots, p$), the following steps are performed. For each expert, $candidate$, having skill s_i, a team is initialized to have $\langle s_i, candidate\rangle$ and its $sumDistance$ is initialized to 0 (lines 5-6). Then, in lines 7-11, for each of the other required skills, s_j, the algorithm finds the expert with s_j that is closest to $candidate$ and adds it (coupled with s_j) to the team. The $sumDistance$ of the team is updated by adding the shortest distance between $candidate$ and the selected expert. After the team has all the required skills (if none of the $C(s_i)$s is empty), if its $sumDistance$ is smaller than that of the current $bestTeam$, it replaces the $bestTeam$ (lines 12-14).

As defined in Section 2, $d(candidate, C(s_j))$ is the shortest distance between $candidate$ and the set of experts $C(s_j)$, which is the shortest distance between $candidate$ and its nearest neighbor in $C(s_j)$. $N(candidate, C(s_j))$ denotes this nearest neighbor. To efficiently find the shortest distance between an expert and a set of experts, the shortest distance between each pair of experts in G, i.e., $d(c_i, c_j)$, has been pre-computed and a hash table is used to store the shortest distances of all pairs for quick access by this algorithm[3]. Thus, the run time of functions $d(candidate, N(s_j))$ and $N(candidate, C(s_j))$ is $O(|C_{max}|)$, where $|C_{max}| = \max|C(s_i)|$ $1 \leq i \leq p$. The **time complexity** of Algorithm 1 is $O(p^2 \times |C_{max}|^2)$, where p is the number of required skills. In the worst case, $|C_{max}| = O(n)$ and the run time of the algorithm is $O(p^2 \times n^2)$, where n is the number of nodes in G. However, in real data sets, $|C_{max}|$ is smaller than n by orders of magnitude [10]. Below we prove that the algorithm finds the best answer with the approximation ratio of two.

Algorithm 1 Finding Best Team without a Leader

Input: graph G, project $P = \{s_1, s_2, \ldots, s_p\}$ and the set of experts with skill s_i, $C(s_i)$, for $i = 1, \ldots, p$.
Output: the best team and its cost

```
1: leastSumDistance ← +∞
2: bestTeam ← ∅
3: for i ← 1 to p do
4:    for every candidate ∈ C(s_i) do
5:       sumDistance ← 0
6:       team ← {⟨s_i, candidate⟩}
7:       for j ← 1 to p and j ≠ i do
8:          distance ← d(candidate, C(s_j))
9:          selectedExpert ← N(candidate, C(s_j))
10:         sumDistance ← sumDistance + distance
11:         team.add(⟨s_j, selectedExpert⟩)
12:      if sumDistance < leastSumDistance then
13:         leastSumDistance ← sumDistance
14:         bestTeam ← team
15: return bestTeam, leastSumDistance
```

THEOREM 2. *Algorithm 1 finds the team of experts that minimizes the sum of distances defined in Definition 2 with 2-approximation.*

PROOF. Consider two teams, one *optimal team* and the team produced by Algorithm 1, denoted here as *best team*[4]. Assume that the number of required skills is p. We denote a node in the *best team* that has the smallest sum of the shortest distances to the other nodes in the team as *candidate node*. Without loss of generality, assume that the *candidate node* is the node related to the first required skill, i.e. s_1. Let's call the shortest distance from the *candidate node* to each of the other nodes in the team $d_{12}, d_{13}, \ldots, d_{1p}$. Thus, the sum of shortest distances from this candidate to all the other nodes in the team is $\sum_{i=2}^{p} d_{1i}$. Let's call this distance the sum of the nearest neighbor distances of the candidate. Based on Algorithm 1, the candidate node has the smallest sum of the nearest neighbor distances among all the nodes in the team and also among all the nodes of all the possible teams including the *optimal team* because Algorithm 1 minimizes such a sum of distances. Let's consider the node with skill s_j in the *optimal team*. Its shortest distance to the node with each of other skills in the *optimal team* is denoted as $o_{1j}, o_{2j}, \ldots, o_{(j-1)j}, o_{j(j+1)}, \ldots, o_{jp}$, respectively. Based on Algorithm 1, the following holds for each j (where $j = 1, \ldots, p$):

$$\sum_{i=1}^{j-1} o_{ij} + \sum_{i=j+1}^{p} o_{ji} \geq \sum_{i=2}^{p} d_{1i} \quad (1)$$

[2]This is due to the triangle inequality: $d(c_i, L) + d(c_j, L) \geq d(c_i, c_j)$, where c_i and c_j are two team members and L is the leader. The proof is omitted due to the space limit.

[3]Note that the pre-computation of the shortest distances was done in [10] as well. If the hash table cannot reside in memory, other indexing techniques such as the one introduced in [6] can be used.

[4]A similar proof is given for the third theorem in our paper [9] for a graph keyword search algorithm.

If we write the above equation for all the p nodes of the *optimal team* and sum up both sides of the inequalities, we have the following equation:

$$2 \times \sum_{i=1}^{p} \sum_{j=i+1}^{p} o_{ij} \geq p \times \sum_{i=2}^{p} d_{1i} \quad (2)$$

The left side of the above equation is twice the communication cost of the *optimal team*. Thus, we have:

$$2 \times (optimal\ cost) \geq p \times \sum_{i=2}^{p} d_{1i} \quad (3)$$

The communication cost of the *best team* is:

$$best\ cost = \sum_{i=1}^{p} \sum_{j=i+1}^{p} d_{ij} = \sum_{i=2}^{p} d_{1i} + \sum_{i=2}^{p} \sum_{j=i+1}^{p} d_{ij} \quad (4)$$

Since the shortest distances satisfy triangle inequality (i.e., $d_{ij} \leq d_{1i} + d_{1j}, i \neq j \neq 1$), the following holds:

$$\sum_{i=2}^{p} d_{1i} + \sum_{i=2}^{p} \sum_{j=i+1}^{p} d_{ij} \leq \sum_{i=2}^{p} d_{1i} + \sum_{i=2}^{p} \sum_{j=i+1}^{p} (d_{1i} + d_{1j}) \quad (5)$$

In the right side of the above equation, each distance d_{1i} appears exactly $p-1$ times. Thus, we have:

$$\sum_{i=2}^{p} d_{1i} + \sum_{i=2}^{p} \sum_{j=i+1}^{p} (d_{1i} + d_{1j}) = (p-1) \times \sum_{i=2}^{p} d_{1i} \quad (6)$$

As a result, we have:

$$best\ cost \leq (p-1) \times \sum_{i=2}^{p} d_{1i} \quad (7)$$

Based on equations 3 and 7, we have:

$$\frac{2 \times (p-1)}{p} \times (optimal\ cost) \geq best\ cost \quad (8)$$

It proves that the sum of the distances of the *best team* is at most twice that of the *optimal team*.

□

4. FINDING BEST TEAM OF EXPERTS WITH A LEADER

To solve Problem 2, we propose the following exact polynomial algorithm. The pseudo code of the algorithm is shown in Algorithm 2. The input of the algorithm is the same as the one for Algorithm 1. The output is the best team of experts with the lowest leader distance ($bestTeam$), its cost ($leastLeaderDistance$) and leader ($bestLeader$). The algorithm works as follows. Each expert in G is a candidate for the leadership. For each node (expert) in graph G, denoted as $leader$, a $team$ and its $leaderDistance$ are initialized in lines 5 and 6. Then, for each of the skills $s_i, 1 \leq i \leq p$, the closest expert to $leader$ with skill s_i and its distance to the $leader$ are identified in lines 8 and 9. The identified expert is added to the team coupled with skill s_i and the $leaderDistance$ is updated accordingly in lines 10-11. After the team has all the required skills, if its $leaderDistance$ is smaller than $leastLeaderDistance$, the value of $leastLeaderDistance$, $bestTeam$ and $bestLeader$ are updated in lines 13-15. The $bestTeam, leastLeaderDistance$ and $bestLeader$ are returned in line 16. In summary, the algorithm selects a leader from all the nodes in G that has the smallest sum of the distances to all sets $C(s_i)$ $(1 \leq i \leq p)$, and forms the team using the expert in each $C(s_i)$ that is closest to the leader.

Since the run time of functions $d(leader, C(s_j))$ and $N(leader, C(s_j))$ is $O(|C_{max}|)$, the **time complexity** of Algorithm 2 is $O(p \times |C_{max}| \times n)$, where p is the number of required skills, $|C_{max}| = \max |C(s_i)|\ 1 \leq i \leq p$ and n is the total number of nodes (experts) in G. One way to reduce the time complexity is to choose the leader from $\bigcup_{i=1}^{p} C(s_i)$. That is, the algorithm does not check all of the nodes in G but only the ones with the required skills. This can be done by adopting Algorithm 1 that finds the team of experts without a leader and returning the *candidate* node in the best team as the leader of the team. However, the team of experts generated in this way may not be an optimal team with respect to the leader distance because it does not check all the possible leaders in G. Similar to the proof of Theorem 2, it can be proved that the approximation ratio of the revised algorithm is two. Due to the space limit, we omit the proof. In the experiments, we use the exact algorithm, i.e., Algorithm 2, for finding the best team of experts with a leader.

Algorithm 2 Finding Best Team with a Leader

Input: graph G, project $P = \{s_1, s_2, \ldots, s_p\}$ and the set of experts with skill s_i, $C(s_i)$, for $i = 1, \ldots, p$.
Output: the best team, its cost and leader

```
1: leastLeaderDistance ← +∞
2: bestTeam ← ∅
3: bestLeader ← ∅
4: for every leader ∈ G do
5:    leaderDistance ← 0
6:    team ← ∅
7:    for i ← 1 to p do
8:       distance ← d(leader, C(s_i))
9:       selectedExpert ← N(leader, C(s_i))
10:      team.add(⟨s_i, selectedExpert⟩)
11:      leaderDistance ← leaderDistance + distance
12:   if leaderDistance < leastLeaderDistance then
13:      leastLeaderDistance ← leaderDistance
14:      bestTeam ← team
15:      bestLeader ← leader
16: return bestTeam, leastLeaderDistance, bestLeader
```

5. ENUMERATING TOP-K TEAMS OF EXPERTS IN POLYNOMIAL DELAY

Sometimes people are interested in finding more than one team with lowest communication costs to form up alternative teams. Since it is infeasible to produce all the teams because the number of possible teams is exponential in the number of required skills, we propose a procedure for producing top-k teams of experts in polynomial delay. Our algorithm for producing a ranked list of results is an adaption of Lawler's procedure [11] for calculating the top-k answers to discrete optimization problems. In Lawler's procedure, the

Table 1: Dividing the search space into disjoint subspaces for finding the teams of experts without a leader.

Subspace	Representative set
SB_0	$\{c_{s_1}\} \times \{c_{s_2}\} \times \{c_{s_3}\} \times \{c_{s_4}\}$
SB_1	$[C(s_1) - \{c_{s_1}\}] \times C(s_2) \times C(s_3) \times C(s_4)$
SB_2	$\{c_{s_1}\} \times [C(s_2) - \{c_{s_2}\}] \times C(s_3) \times C(s_4)$
SB_3	$\{c_{s_1}\} \times \{c_{s_2}\} \times [C(s_3) - \{c_{s_3}\}] \times C(s_4)$
SB_4	$\{c_{s_1}\} \times \{c_{s_2}\} \times \{c_{s_3}\} \times [C(s_4) - \{c_{s_4}\}]$

search space is first divided into disjoint sub-spaces; then the best answer in each subspace is found and used to produce the current global best answer. The sub-space that produces the best global answer is further divided into sub-subspaces and the best answer among its sub-subspaces is used to compete with the best answers in other sub-spaces in the previous level to find the next best global answer. Two main issues in this procedure are how to divide a space into disjoint subspaces and how to find the best answer within a (sub)space. Lawler proved that if these two issues are solved in polynomial time, the procedure generates ranked list of answers in polynomial delay [11].

5.1 Finding Top-k Teams without a Leader

We first informally describe the idea for dividing the search space into disjoint subsets using an example[5]. Suppose that the project consists of four skills, i.e., $\{s_1, s_2, s_3, s_4\}$, and we want to find the top-k teams of experts without a leader. Let $C(s_i)$ be the set of experts in graph G that have skill s_i. Thus, the search space that contains the best answer can be represented as $C(s_1) \times C(s_2) \times C(s_3) \times C(s_4)$. From this space, we can use Algorithm 1 for finding the best (approximate) team in polynomial time. Assume that the best answer is $(c_{s_1}, c_{s_2}, c_{s_3}, c_{s_4})$, where c_{s_i} is a node (expert) in graph G which has skill s_i. Based on this best answer, the search space is divided into 5 subspaces SB_0, SB_1, SB_2, SB_3 and SB_4 as shown in Table 1, where SB_0 contains only the best answer. It should be noted that the subspaces are **disjoint** and their union covers the whole search space.

After finding the best answer and dividing the search space into disjoint subsets, the best answer in each subspace except SB_0 is found using Algorithm 1. These best answers are inserted into a priority queue, where the answers are ranked in ascending order according to their communication cost. Obviously, the second best answer is the one at the top of the priority queue. Then, the top answer is returned and removed from the queue, its corresponding space is divided into subspaces and the best answer (if any) in each new subspace is added to the priority queue. This procedure continues until the priority queue becomes empty or the required number of teams, i.e. k, is returned by the algorithm.

The pseudo code of algorithm that generates top-k teams without a leader is presented in Algorithm 3. The main body of the algorithm is similar to other polynomial delay algorithms discussed in [7, 13]. It is modified to perform in the setting of producing ranked teams of experts from a graph. In line 2, procedure $FindBestTeam$ is called to find the best answer in space W in polynomial time. This procedure can be Algorithm 1 or one of the two algorithms introduced in [10], all having polynomial time complexity. It should be noted that W is the whole search space that contains all of the experts with their skills. Thus, the first best answer should be found in this space. If the best answer exists (i.e., A is not empty), A, together with its cost and related space W, is inserted into $Queue$ in line 4. The $Queue$ is maintained in the way that its elements are ordered in ascending order according to their communication costs. The while loop starting at line 5 is executed until the $Queue$ becomes empty or k answers have been outputted. In line 6, the top of the $Queue$ is removed, which contains the best answer in the $Queue$ and the space that this answer is produced from. We assign this space to V and the best answer to A. The answer in A is outputted in line 7. If the number of answers has reached k, the algorithm terminates in line 10. In line 11, procedure $ProduceSubSpaces$ produces p new subspaces based on the current answer A and current search space V. These subspaces are shown by SV_i. In lines 12-15, for each of these new subspaces, the best answer is found and inserted into the $Queue$ with its cost and related subspace. As stated before, procedure $FindBestTeam$ terminates in polynomial time. Thus, if procedure $ProduceSubSpaces$ terminates in polynomial time, then Algorithm 3 produces answers in polynomial delay. The pseudo code of procedure $ProduceSubSpaces$ is presented in Algorithm 4. It takes the best answer of the previous step, i.e. A, and the search space of previous step, i.e. V, as input. It produces p disjoint subspaces, $\langle SV_1, \ldots, SV_p \rangle$, as output. In this procedure, SV_i^j specifies the j-th element of SV_i. It is a polynomial procedure and runs in $O(p^2)$.

Algorithm 3 Algorithm for Generating Top-k Teams with a Leader in Polynomial Delay

Input: graph G; project $P = \{s_1, s_2, \ldots, s_p\}$; number of required teams k and the sets of experts each with a required skill $W = \{C(s_1), C(s_2), \ldots, C(s_p)\}$
Output: the set of top-k ordered teams of experts printed with polynomial delay

```
1: Queue ← an empty priority queue
2: ⟨A, cost⟩ ← FindBestTeam(G, T, W)
3: if A ≠ ∅ then
4:    Queue.insert(⟨A, cost, W⟩)
5: while Queue ≠ ∅ do
6:    ⟨A, V⟩ ← Queue.removeTop()
7:    print(A)
8:    k ← k − 1
9:    if k = 0 then
10:      return
11:   ⟨SV1, SV2, ..., SVp⟩ ← ProduceSubSpaces(A, V)
12:   for i ← 1 to p do
13:      ⟨Ai, costi⟩ ← FindBestTeam(G, T, SVi)
14:      if Ai ≠ ∅ then
15:         Queue.insert(⟨Ai, costi, SVi⟩)
```

5.2 Finding Top-k Teams with a Leader

The procedure for finding the top-k teams with a leader is similar to Algorithm 3. But since the search space for the best team is $C(s_1) \times C(s_2) \cdots \times C(s_p) \times C$, where C is the set of experts in G from which the leader is chosen, the search space is divided into $p + 2$ subspaces after the best answer is found (instead of $p + 1$ subspaces as in Algorithm 3). Table 2 illustrates the subspaces divided from the search

[5]Our approach for dividing a search space is similar to the idea used in [13].

Table 2: Dividing the search space into disjoint subspaces for finding the teams of experts with a leader. C is the set of experts in G. The leader of the best answer $(c_{s_1}, c_{s_2}, c_{s_3}, c_{s_4})$ is L_1.

Subspace	Representative set
SB_0	$\{c_{s_1}\} \times \{c_{s_2}\} \times \{c_{s_3}\} \times \{c_{s_4}\} \times \{L_1\}$
SB_1	$[C(s_1) - \{c_{s_1}\}] \times C(s_2) \times C(s_3) \times C(s_4) \times C$
SB_2	$\{c_{s_1}\} \times [C(s_2) - \{c_{s_2}\}] \times C(s_3) \times C(s_4) \times C$
SB_3	$\{c_{s_1}\} \times \{c_{s_2}\} \times [C(s_3) - \{c_{s_3}\}] \times C(s_4) \times C$
SB_4	$\{c_{s_1}\} \times \{c_{s_2}\} \times \{c_{s_3}\} \times [C(s_4) - \{c_{s_4}\}] \times C$
SB_5	$C(s_1) \times C(s_2) \times C(s_3) \times C(s_4) \times [C - \{L_1\}]$

space after the first best team with a leader is found for the same example in Table 1. Both Algorithms 3 and 4 are changed to reflect the extension of the search space and difference in partitioning the search space. For example. *ProduceSubSpaces* takes also the leader of the best team and the previous search space for the leader as input. It produces $p+1$ (instead of p as in Algorithm 4) subspaces. The *FindBestTeam* procedure should be Algorithm 2 and returns a leader in addition to the best team and its cost. Due to the space limit, we omit further details.

Algorithm 4 Algorithm for Producing Sub Spaces

Input: the best answer of previous step $A = \langle c_1, c_2, \ldots c_p \rangle$ and set of experts which have the required skills $V = \{C(s_1), C(s_2), \ldots, C(s_p)\}$
Output: p new subspaces $\langle SV_1, \ldots, SV_p \rangle$

1: **for** $i \leftarrow 1$ **to** p **do**
2: **for** $j \leftarrow 1$ **to** $i-1$ **do**
3: $SV_i^j \leftarrow \{c_j\}$
4: $SV_i^i \leftarrow V_i - \{c_i\}$
5: **for** $j \leftarrow i+1$ **to** p **do**
6: $SV_i^j \leftarrow V_j$
7: **return** $\langle SV_1, \ldots, SV_p \rangle$, where $SV_i = SV_i^1 \times \cdots \times SV_i^p$

6. EMPIRICAL EVALUATION

In this section, we refer to the approximation Algorithm 1 as BEST-SUMDISTANCE and Algorithm 2 as BEST-LEADER. For the purpose of comparison, we implemented two algorithms introduced in [10]. The first algorithm that minimizes the diameter of the team is referred to as RAREST-FIRST and the second one that minimizes the weight of the MST of the team is referred to as ENHANCED-STEINER [10]. All the algorithms are implemented in Java. The experiments are conducted on an Intel(R) Core(TM) i7 2.80GHz computer with 4GB of RAM.

6.1 The Datasets

The DBLP and IMDb data sets are used in our experiments. The DBLP graph is produced from the DBLP XML data[6] taken on June 24, 2010. The dataset contains information about a collection of papers and their authors. The set of experts and skills are generated in the same way as in [10] as follows. We only keep the papers of some major conferences in computer science: Data Base = {SIGMOD, VLDB, ICDE, ICDT, EDBT, PODS}, Data Mining = {KDD, WWW , SDM, PKDD, ICDM}, Artificial Intelligence = {ICML, ECML, COLT, UAI} and Theory = {SODA, FOCS, STOC, STACS}. The set of experts consists of authors that have at least three papers in the DBLP. The skills of each expert is the set of terms that appear in the titles of at least two publications of the expert. Two experts are connected together if they have at least two papers together. The weight of the edge between two nodes n_i and n_j is equal to $1 - |\frac{p_{n_i} \cap p_{n_j}}{p_{n_i} \cup p_{n_j}}|$ where p_{n_i} is the set of papers of author (node) n_i. The final graph has 5,658 nodes (experts) and 8,588 edges.

The part of the IMDb dataset used in our experiments contains information about the actors and the list of movies that each actor played in[7]. The titles of the movies and the names of the actors are specified in the dataset. We consider as experts the actors who have played at least eight movies from 2000 to 2002. The skills of each actor is the set of terms that appear in the title of a movie of the actor. Two actors are connected together if they played in at least four movies together. The weight of an edge is determined in the same way as for DBLP. The final graph has 6,784 nodes and 35,875 edges. Clearly, the graph of IMDb is denser than that of DBLP. We do not use movie genres as skills because the number of genres in IMDb is only 20 and many actors have played in a great portion of genres, which leads to generation of top-100 teams each with only one member since many actors have played in all the required genres. Although using terms in movie titles as skills may not be meaningful, the data set is still good to use for the purpose of comparing the algorithms in some performance measures.

6.2 Experimental Setup

The *projects* used in the experiments are generated as follows. Each project is specified by a set of skills. We allow the number of skills in a project, i.e., p, to vary from 4 to 10. For each value of p, we randomly generate 100 projects and compute the average result obtained by each algorithm. Skills have different frequencies. The frequency of a skill is the percentage of experts in the social network that possess the skill. The frequency of the skills in our experiments varies from 0.5 to 5.0 percent. Other values of skill frequency lead to similar conclusions. We run the top-k procedures proposed in Section 5 using 4 different algorithms as the *FindBestTeam* procedure (i.e., BEST-SUMDISTANCE, BEST-LEADER, RAREST-FIRST, or ENHANCED-STEINER). We experimented with different values of k from 1 to 100. When it is not varied, the value of k is set to 50 by default. Also, the same as in [10], if, for a project, the team produced by at least one of the algorithms does not lead to a connected graph, the result of this project is ignored.

6.3 Performance Measures

We use two sets of performance measures to evaluate the quality of the teams produced by different algorithms. The first set contains three communication cost measures: (1) the diameter of the team, which is the largest shortest distance between any two nodes in the team, (2) the cost of the minimum spanning tree (MST) on the subgraph of the team and (3) the sum of distances measure defined in this paper. Recall that RAREST-FIRST is a greedy algorithm that aims at minimizing the diameter of the team, ENHANCED-STEINER is an approximation algorithm that tries to minimize MST, and BEST-SUMDISTANCE is an approximation

[6] http://dblp.uni-trier.de/xml/

[7] http://www.imdb.com/interfaces

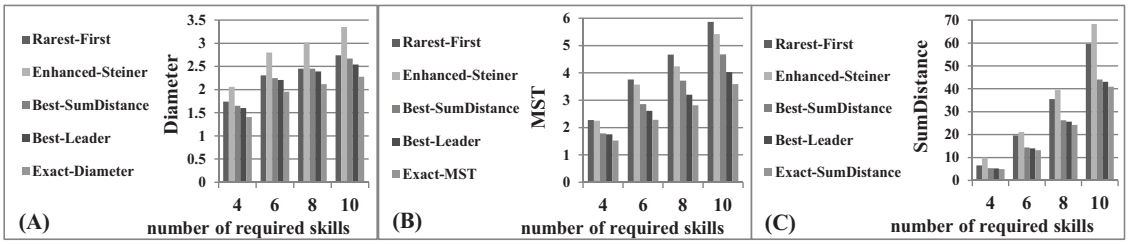

Figure 2: Results on the DBLP graph Using the First Set of Performance Measures.

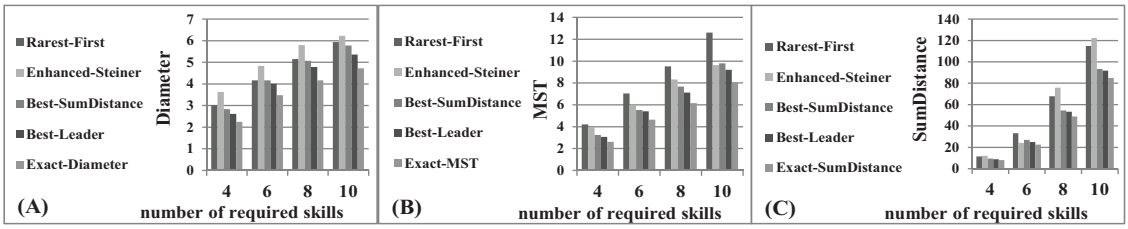

Figure 3: Results on the IMDb graph Using the First Set of Performance Measures.

algorithm for minimizing the sum of distances. The purpose of using these performance measures is to see how well each algorithm minimizes each of these cost measures in comparison to other algorithms.

The second set of performance measures is for evaluating the team quality on the DBLP data set. It contains three metrics. The first one is the cardinality of the team. To reduce the communication cost of a project, we usually prefer teams with smaller cardinality [10]. The second metric is the number of common publications shared by at least two experts in the team, which is the number of the papers in DBLP co-authored by at least two experts in the team. The more common publications the experts have, the higher chance that they can communicate well with each other. The third metric is the skill count, defined as the average number of times that a skill appears in the list of papers of an expert in the team, averaged over all the skills required in the project and over all the experts in the team. The higher the skill count, the more expertise the team has with respect to the set of skills in the project. The purpose of using these performance measures is to look at the quality of resulting teams from aspects different than communication costs.

6.4 Results on the First Set of Performance Measures

Figures 2 and 3 show the results of the four algorithms in terms of the diameter, MST, and sum of distances costs of the top ranking teams on the DBLP and IMDb datasets respectively. Each bar represents the average cost of the top-50 answers from an algorithm over 100 randomly generated projects for a specific number of required skills. In each subgraph, we also include the results of an exact algorithm that minimizes the cost metric used as the performance measure. For example, the results of the exact algorithm that minimizes the diameter of the team on the DBLP data set is shown in Figure 2 (A) and is named as Exact-Diameter. Note that finding a team of experts that minimizes the diameter or MST is an NP-hard problem [10]. We proved that finding teams of experts which minimizes the sum of the distances is also NP-hard. Thus, it is not possible to find the best team of experts that minimizes any of these measures in polynomial time. The exact algorithms use exhaustive search to find all the possible teams for each project, rank the teams based on their respective cost measure, and output the top ranking teams. The purpose of showing the exact algorithm results is to provide the ground truth for each measure and see how close each approximation algorithm performs to the ground truth.

As shown in the results, in almost all of the experiments, algorithms Best-SumDistance and Best-Leader outperforms Rarest-First and Enhanced-Steiner on all of the three cost measures. Their results are very close to the optimal answers produced by the exact algorithms whose performance is the best in each subgraph as expected. Note that although Best-SumDistance and Best-Leader do not minimize the diameter or the MST cost of the teams, they produce better teams with lower diameter and MST costs than Rarest-First and Enhanced-Steiner. This is due to the search strategy used in the Rarest-First and Enhanced-Steiner approximation algorithms: Rarest-First only checks the nodes having the skill with the lowest cardinality support [10], and Enhanced-Steiner starts with a random node from the set of skill holders [10]. Both algorithms might miss some good teams.

Between Best-Leader and Best-SumDistance, the results of Best-Leader are a bit better. This is because Best-Leader evaluates all nodes in the graph for finding the best leader. Comparing the results of Best-SumDistance to those of Exact-SumDistance in subgraph (C) of Figure 2 or 3, we observe that although in theory the sum of distances of an answer from Best-SumDistance can be twice that of the corresponding optimal answer, the difference is small in practice (less than 10% in the worst case). This indicates that the quality of the proposed approximation algorithm is high.

6.5 Results on the Second Set of Performance Measures

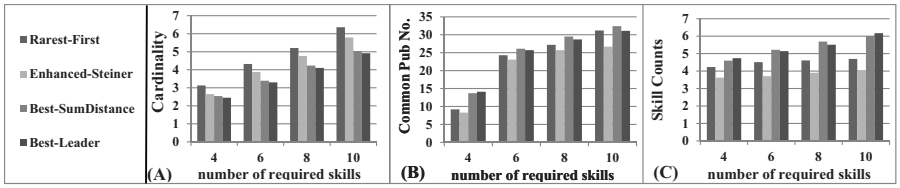

Figure 4: Results of Four Algorithms on DBLP Using the Second Set of Performance Metrics

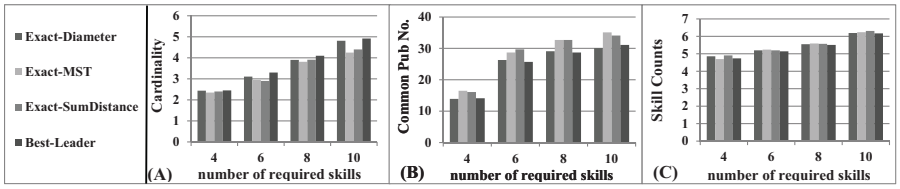

Figure 5: Comparison of Exact Algorithms on DBLP Using the Second Set of Performance Metrics.

Figure 4 compares the four algorithms in terms of the cardinality, the number of common publications and the skill count of the team. The results show that BEST-SUMDISTANCE and BEST-LEADER outperforms RAREST-FIRST and ENHANCED-STEINER on all these metrics, that is, they produce smaller teams with higher number of shared publications and more expertise in the required skills. The results of BEST-SUMDISTANCE and BEST-LEADER are very close to each other. Although our proposed algorithms do not explicitly optimize these metrics, they generate teams with higher quality than two other algorithms with respect to these metrics.

In order to see which cost function (diameter, MST, sum of distances and leader distance) is the best to optimize regardless of the search strategies (i.e., the approximation algorithms designed for them), we present in Figure 5 the results of the four **exact** algorithms in terms of the team cardinality, common publication number and skill count. Note that the BEST-LEADER algorithm is an exact algorithm for minimizing the leader distance of a team. The results show that the MST and sum of distances functions have very similar performance in all the three metrics. Both of them are better than the diameter and leader distance functions in terms of team cardinality and common publication numbers. The four functions perform almost the same on the skill count. The reason why the leader distance leads to a bit larger team and fewer common publications than the sum of distances and MST is that the selected leader may be an "extra" member without a required skill.

Since it is not possible to have polynomial exact algorithms that minimize diameter, MST or sum of distances, how to design the approximation algorithm is important. The results in Figures 2, 3 and 4 indicate that our proposed polynomial algorithms (i.e., BEST-SUMDISTANCE and BEST-LEADER) for finding teams of experts with or without a leader outperform the existing **approximation** algorithms that minimize the diameter or MST function.

6.6 Scalability

To test the scalability of the proposed algorithms, we generate DBLP graphs of different sizes by using different thresholds for the number of papers an expert has authored to determine whether a node (i.e., expert) should be included in the graph. The threshold is set to 6, 5, 4 and 3 to generate graphs of 4 different sizes ranging from 2.2k to 5.6k nodes. Figure 6 (A) illustrates how the average run time of algorithm BEST-SUMDISTANCE for producing one team changes with the number of nodes in the DBLP graph for different numbers of required skills. Although in the worst case (when all the experts in the graph possess a required skill), the run time is quadratic with respect to the number of nodes (as discussed in Section 3), in practice this rarely happens and we observe that the run time increases (almost) linearly when the number of nodes increases. Figure 6 (B) shows how the average time for producing one team (which is the total run time divided by k) changes with k, i.e., the number of top teams produced. Obviously, the time does not change with the value of k, which indicates that the total run time of our top-k algorithm grows linearly when k increases. This indicates that BEST-SUMDISTANCE and our top-k procedure scale well with the number of nodes and k values. Due to the space limit, the results of BEST-LEADER are not presented. But they showed similar trends.

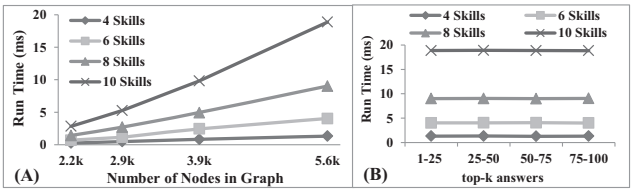

Figure 6: Scalability of Best-SumDistance on DBLP.

6.7 A Sample Result

We compare the algorithms through an example. We form a project by randomly selecting 4 keywords as the required skills from the paper titles listed on the "Awards" page of the CIKM 2010 web site[8]. The keywords are *multilingual*, *interface*, *ranking* and *constraints*. The best team formed by each algorithm is shown in Table 3. The common publication number (denoted as *Pub. No*) of both BEST-SUMDISTANCE

[8]http://www.yorku.ca/cikm10/awards.php

Table 3: Best teams for project requiring "skills": *multilingual*, *interface*, *ranking* and *constraints*

Algorithm	*Multilingual*	*Interface*	*Ranking*	*Constraints*	Leader	Pub. No.	*Diameter*	*MST*	*SumDist*
Rarest-First	*A. Kumaran*	*H. Jagadish*	*M. Ramanath*	*W. Fan*	–	0	3.6	5.4	14.6
Enhanced-Steiner	*A. Kumaran*	*H. Jagadish*	*M. Ramanath*	*W. Fan*	–	0	3.6	5.4	14.6
Best-SumDistance	*A. Kumaran*	*R. Agrawal*	*R. Agrawal*	*K. Shim*	–	3	3.6	4.5	12.9
Best-Leader	*A. Kumaran*	*R. Agrawal*	*R. Agrawal*	*K. Shim*	R. Agrawal	3	3.6	4.5	12.9

and Best-Leader is 3 while it is 0 for Rarest-First and Enhanced-Steiner. Also, Best-SumDistance and Best-Leader produce teams with less MST and sum of distances costs than Rarest-First and Enhanced-Steiner. All the teams have the same diameter in this example.

7. RELATED WORK

Discovering a team of experts from a social network has been introduced in [10]. Authors of [12] generalize this problem by associating each required skill with a specific number of experts. However, they do not provide an approximation ratio for their heuristic algorithms. Authors of [1] proposed methods for finding teams of experts in which the load of the experts are balanced in the presence of several tasks. They do not consider finding teams with low communication cost. In this work, we extend the work of [10] in several directions as follows. Two new communication cost functions are proposed. An approximation algorithm which find the best team of experts is proposed. The problem of finding a team of experts with a leader is defined. A procedure for producing top-k teams of experts is also designed.

The team formation problem has been studied in the operation research community. The authors use branch and bound, genetic algorithms and simulated annealing to solve the problem [4, 14, 5]. The authors of [3] study the dynamics of group formation procedures and how they affect on the formation of groups in the network. In a later work, authors discuss this problem from a game-theoretic point of view [8]. The basic difference between our work and the works in operation research is that they do not assume the existing of a social network behind the experts.

The problem of keyword search in graphs takes a considerable amount of attention in the database community in recent years [7, 13, 9]. Each node in the graph consists of some keywords. The purpose is to find a subgraph that covers the input keywords and whose nodes are close to each other, which is similar to the objective of team formation in social networks that minimizes the team communication cost. The *leader distance* function defined in this paper has not been used in either team formation or keyword search. Also, in algorithms for keyword search over graphs, a threshold is usually set for the maximum distance between two nodes in a final answer [13], but it is not used here.

8. CONCLUSION

We studied the problem of discovering a team of experts from a social network that minimizes the communication cost among team members. We defined two communication cost functions and proposed an approximation algorithm for finding the best team of experts without a leader that minimizes the sum of distance function. We proved that the approximation ratio of the algorithm is 2. We also defined the problem of finding a team of experts with a leader and proposed an exact polynomial algorithm to solve this problem. In addition, we proposed procedures for producing top-k teams of experts with or without an expert in polynomial delay. Our experiments on two real data sets showed that the proposed algorithms produce teams with lower communication costs than two existing approximation algorithms. We also showed that our algorithms produce smaller teams with more common publications and more expertise in the required skills than the two existing approximation algorithms on the DBLP dataset. In the future, we will explore how to incorporate constraints in team formation in social networks.

9. REFERENCES

[1] A. Anagnostopoulos, L. Becchetti, C. Castillo, A. Gionis, and S. Leonardi. Power in unity: Forming teams in large-scale community systems. In *Proc. of CIKM'10*, 2010.

[2] E. M. Arkin and R. Hassin. Minimum-diameter covering problems. *Networks*, 36, 2000.

[3] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan. Group formation in large social networks: membership, growth, and evolution. In *Proc. of KDD'06*, 2006.

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

[5] E. Fitzpatrick and R. Askin. Forming effective worker teams with multi functional skill requirements. *Comput. Ind. Eng.*, 48(3):593–608, 2005.

[6] R. Goldman, N. Shivakumar, S. Venkatasubramanian, and H. Garcia-Molina. Proximity search in databases. In *Proc. of VLDB'98*, 1998.

[7] K. Golenberg, B. Kimelfeld, and Y. Sagiv. Keyword proximity search in complex data graphs. In *Proc. of SIGMOD'08*, 2008.

[8] M. Jackson. *Network formation*. The New Palgrave Dictionary of Economics and the Law, 2008.

[9] M. Kargar and A. An. Keyword search in graphs: Finding r-cliques. In *Proc. of VLDB'11*, 2011.

[10] T. Lappas, L. Liu, and E. Terzi. Finding a team of experts in social networks. In *Proc. of KDD'09*, 2009.

[11] E. Lawler. A procedure for computing the k best solutions to discrete optimization problems and its application to the shortest path problem. *Management Science*, 18, 1972.

[12] C. Li and M. Shan. Team formation for generalized tasks in expertise social networks. In *Proc. of IEEE International Conference on Social Computing*, 2010.

[13] L. Qin, J. Yu, L. Chang, and Y. Tao. Querying communities in relational databases. In *Proc. of ICDE'09*, 2009.

[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.