J Supercomput (2013) 66:721–737
DOI 10.1007/s11227-013-0907-x

Efficient algorithms for team formation with a leader in social networks

Ming-Chin Juang · Chen-Che Huang · Jiun-Long Huang

Published online: 23 March 2013

Abstract Given a project with a set of required skills, it is an important and challenging problem of find a team of experts that have not only the required skill set but also the minimal communication cost. Furthermore, in view of the benefits of greater leaders, prior work presented the team formation problem with a leader where the leader is responsible for coordinating and managing the project. To find the best leader and the corresponding team, the prior work exhaustively evaluates each candidate and the associated team, incurring substantial computational cost. In this paper, we propose two efficient algorithms, namely the BCPruning algorithm and the SSPruning algorithm, to accelerate the discovery of the best leader and the corresponding team by reducing the search space of team formation for candidates. The BCPruning algorithm aims at selecting better initial leader candidates to obtain lower communication cost, enabling effective candidate pruning. On the other hand, the SSPruning algorithm allows each leader candidate to have a lower bound on the communication cost, leading some candidates to be safely pruned without any computation. Besides, the SSPruning algorithm exploits the exchanged information among experts to aid initial candidate selection as well as team member search. For performance evaluation, we conduct experiments using a real dataset. The experimental results show that the proposed BCPruning and SSPruning algorithms are respectively 1.42–1.68 and 2.64–3.25 times faster than the prior work. Moreover, the results indicate that the proposed algorithms are more scalable than the prior work.

Keywords Social network · Social intelligence · Team formation

M.-C. Juang · C.-C. Huang · J.-L. Huang (✉)
Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan, ROC
e-mail: jlhuang@cs.nctu.edu.tw

M.-C. Juang
e-mail: vder.cs99g@nctu.edu.tw

C.-C. Huang
e-mail: cchuang.cs95g@nctu.edu.tw

1 Introduction

A project is usually associated with a set of the required skills. To accomplish a project, it is necessary to discover a team of experts that together possess the required skills. However, the success of a project is not simply dependent on the required expertise of the experts in the team. A more crucial factor is whether the experts can communicate and collaborate effectively. Hence, given a project, it is necessary and desirable to select a team of experts where the required skill set is covered by the skills of the experts and the communication cost of the team is minimal.

Lappas et al. [12] were the first to define the communication cost of a team of experts in the presence of the social network of the experts. They presented two cost functions, namely the *diameter communication cost* and *minimum spanning tree cost*, to evaluate the communication effectiveness of a team. The former is defined as the diameter of the subgraph formed by the team of the experts, while the latter is the cost of minimal spanning tree on the subgraph. These two cost functions are unstable since a radical change in the solution may occur with a slight change in the graph. To overcome this problem, Kargar and An [11] proposed a new cost function, which is called the *sum of distances* and measures the communication cost of a team by summing up the shortest distances between the experts for each pair of skills.

In addition, Kargar and An introduced the team formation with a leader where the leader is in charge of managing and coordinating the team. The communication cost of the team and the leader, called the *leader distance*, is measured as the sum of shortest distances between the leader and the corresponding expert for each required skill. Since a good leader is usually able to lead a team to work more efficiently and effectively, it is beneficial to form a team with a great leader. For example, consider the social network of 6 experts $\{A, B, C, D, E, F\}$ in Fig. 1. The number next to an edge is the communication cost between the two experts. Suppose that, for a given project, the team composed of experts $\{B, C, E, F\}$ can cover the required skill set for the project. With expert C as the leader, the leader distance (communication cost) of the team equals to 2.7 $(1.5 + 0 + 0.5 + 0.7)$ and is minimal. However, the leader distance can be further reduced to 2.3 $(1.3 + 0.2 + 0.3 + 0.5)$ and the team can collaborate more effectively if a great leader D is employed to supervise and coordinate the team members. To discover the best leader and the corresponding team of experts, Kargar and An presented a brute-force algorithm that considers each expert as a leader candidate and searches the best team for the candidate. Their algorithm determines the best leader and the corresponding team of experts after all the experts are evaluated, suffering from significant computational cost.

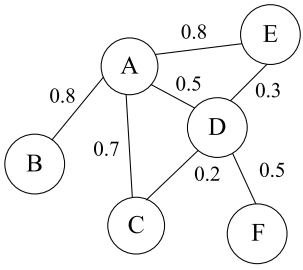


Fig. 1 The benefit of team formation with a leader

In this paper, we propose two efficient algorithms to identify the team of experts with a leader that not only fulfill the skill requirement of a project but also have the minimum leader distance. The two proposed algorithms, namely the *BCPruning* algorithm and the *SSPruning* algorithm, aim at reducing the search space of team formation for each leader candidate. The BCPruning algorithm selects better initial leader candidates that are on multiple shortest paths between the experts to obtain a lower communication cost earlier, enabling the early termination of searching team members for candidates. On the other hand, the SSPruning algorithm requires each expert to spread its possessing skills to neighboring experts within a specific distance. With the skill information, each expert can compute a lower bound on the leader distance with respect to a given project, leading to safe pruning of some candidates. In addition, the skill information is beneficial for initial leader selection and team member search, resulting in a reduction in search space for candidates. To evaluate the performance of the proposed algorithms, we conduct experiments on the real DBLP dataset and compare the proposed algorithms with the prior work [11]. The experimental results demonstrate that the proposed BCPruning and SSPruning algorithms are respectively 1.42–1.68 and 2.64–3.25 times faster than the prior work. The experimental results also show that the BCPruning and SSPruning algorithms are more scalable than the prior work.

The remainder of this paper is organized as follows. Section 2 reviews related work. The problem formulation is presented in Sect. 3. In Sect. 4, we elaborate the proposed BCPruning and SSPruning algorithms. We report the experimental results in Sect. 5. Finally, Sect. 6 concludes this paper and discusses future work.

2 Related work

The team formation problem is crucial in any organization and thus has been well studied in the field of operations research. In [19], Zakarian and Kusiak developed a methodology for team formation based on the analytical hierarchical process and the quality function development method. Besides, they formulated the team formation problem as an integer linear program (ILP) and solved it using the branch-and-cut technique. Fitzpatrick and Askin [7] evaluated the quality of a team by measuring individuals' drive and temperament based on the Kolbe Conative Index. Chen and Lin [6] developed a quantitative scheme to represent the multifunctional knowledge, teamwork capability, worker relationship of team members. With the developed scheme, a procedure was introduce to find the best team. In [4], Baykasoglu et al. proposed using a fuzzy optimization model for the team formation and solving the fuzzy model by devising a simulated annealing algorithm. In [18], Wi et al. proposed to evaluate the personal knowledge and the familiarity of personnel via a fuzzy interference system. The team formation problem was modeled as an integer programming problem and solved by employing a genetic algorithm. Gaston et al. [9] took into account the network structure between individuals and studied the impact of the network structures on the team performance. In [3], Backstrom investigated the dynamics of group-formation procedures and the effect on group formation. Such an issue was discussed by Jackson [10] from the game-theoretic perspective.

Recently, social networks [5, 15] have been an active research field. Through Information Retrieval [16, 17], we can construct social networks from collected data. In view of the fact that the above studies addressed team formation without considering the social network of individuals, Lappas et al. [12] presented the first work on this problem in the presence of the social network. They proposed two communication cost functions for a team, namely the *diameter communication cost (Cc-R)* and *minimum spanning tree (MST) communication cost (Cc-MST)*. Besides, they proved the NP-completeness of the team formation problem with minimal Cc-R or Cc-MST and presented appropriate approximate algorithms for two instances of the problem. In [13], Li and Shan generalized the team formation problem by enabling each required skill to have a specific number of experts. They improved the team effectiveness and team formation efficiency by devising a density-based measure and a grouping-based approach, respectively. Anagnostopoulos et al. [1] considered the load balancing among the experts in the presence of multiple tasks and designed the algorithms to solve this problem. Based on their previous work [1], Anagnostopoulos et al. [2] proposed algorithms to simultaneously achieve minimal communication cost and load balancing for team formation. Majumder et al. [14] also considered the loading of experts in the formed team but focused on preventing the loading of each expert from exceeding the capacity of the expert. Arguing the instability and insensitivity of the cost functions of [12], Kargar and An [11] introduced two new cost functions, the *sum of distances* and the *leader distance*. Moreover, they considered that a team of experts may need a leader for project coordination and management. They also developed an algorithm to allow selecting top-k teams instead of one single team.

3 Problem formulation

Although the problem we intend to address in this paper is the same as in [11], we present the problem formulation in this section to make this paper self-contained. Let $S = \{s_1, s_2, \ldots, s_m\}$ and $X = \{x_1, x_2, \ldots, x_n\}$ denote a set of m skills and a set of n experts, respectively. The skill set of expert x_i is denoted by $S(x_i) \subseteq S$. Expert x_i is said to possess a skill s_j if $s_j \in S(x_i)$. For each skill s_j, $T(s_j) = \{x_i | s_j \in S(x_i)\}$ represents the set of the experts having s_j. A project $P = \{s_1, s_2, \ldots, s_p\} \subseteq S$ is composed of the skills required to complete the project. Given a project P, a subset of experts C' is said to be able to *cover* P if $\forall s_j \in P\ \exists x_i \in C',\ s_j \in S(x_i)$. Note that we say that a subset of experts $C' \subset X$ have skill s_j if at least one of them possesses s_j.

The social network of experts is modeled as an undirected and weighted graph $G = (X, E)$ where each vertex is an expert $x \in X$ and E is the set of edges connecting the vertices. An edge $e_{ij} \in E$ exists if two experts x_i and t_j have collaborated with each other. The weight of e_{ij} represents the communication cost between two experts x_i and x_j. The more two experts work together, the lower the communication cost (weight) between the two experts. The distance between two experts x_i and x_j, denoted by $dist(x_i, x_j)$, is the sum of the weights of the edges on the shortest path between x_i and x_j. Similar to [11], if x_i and x_j are disconnected, $dist(x_i, x_j)$ is set to a value larger than the sum of all pairwise shortest distances between all pairs of vertices in G.

Definition 1 (Leader distance) Given a project $P = \{s_1, s_2, \ldots, s_p\}$, a team T of experts for P is $T = \{\langle s_1, x_{s1}\rangle, \langle s_2, x_{s2}\rangle, \ldots, \langle s_p, x_{sp}\rangle\}$ where each pair $\langle s_i, x_{si}\rangle$ means that $x_{si} \in X$ and $s_i \in S(x_{si})$. Assume that team T has a leader $L \in X$ where L may not be one of the team members in T. The leader distance of T with leader L is defined as $leaderDistance = \sum_{i=1}^{p} dist(x_{si}, L)$.

Definition 2 (Team formation with a leader) Given a graph G of the social network composed of experts X, the problem of team formation with a leader for a project P is to return a team T of experts and an expert L as the leader with the minimal leader distance.

4 Proposed algorithms

In this section, we elaborate on the two proposed algorithms, namely the *BCPruning algorithm* and the *SSPruning algorithm*. The BCPruning algorithm attempts to initially select promising leader candidates to obtain smaller leader distances, resulting in reduced search space of team formation for leader candidates. On the other hand, the SSPruning algorithm presents skill information exchange among experts that enables each leader candidate to compute a lower bound on the leader distance, leading some candidates to be excluded directly without any computation. Besides, the information exchange among experts is helpful for the initial candidate selection as well as team member search. In what follows, we describe the two proposed algorithms in detail.

4.1 Betweenness centrality: BCPruning algorithm

As mentioned earlier, to find the best team of experts with a leader, the prior work [11] considers each expert as a leader candidate. With respect to each required skill, a leader candidate searches the closest expert (with the shortest distance). The prior work identifies the best leader and the corresponding team after evaluating all the leader candidates. However, utilizing the currently smallest leader distance is likely to exclude a candidate as the desired leader before coping with all the required skills. For example, assume that the currently smallest leader distance is 1.2. Consider that the accumulated communication cost for a leader candidate and found team members is 1.35 and several required skills remain not covered. In this case, this candidate is guaranteed not to be the desired leader and thus can be safely skipped without further computation for the remaining skills. Based on this observation, we design the BCPruning algorithm to attempt to select better initial leader candidates with smaller leader distances, reducing the computation of team member search for subsequent candidates.

4.1.1 Betweenness centrality calculation

Since the leader distance is determined by the communication cost between the team member corresponding to each required skill and the leader, we utilize the *betweenness centrality* [8] as the criterion for leader candidate selection.

Algorithm 1: BCinDijkstra algorithm

```
Input: social network: G(X, E)
Output: distance dist(v, t), betweenness centrality of all experts BC(v)
Initialize betweenness centrality of all experts to 0 and dist(v, t) as ∞
for each expert x ∈ X do
    Set dist(v, v) to 0
    Set unvisited expert set UX = X
    while UX is not empty do
        Select the expert v with the smallest distance dist(x, v) from UX and remove v from UX
        if dist(x, v) is ∞ then
            break
        end
        Increment the BC of the nodes on the path from x to v by the fraction of the shortest paths
        that pass through the node
        for each neighbor u of v do
            d = dist(x, v) + dist(v, u) if d < dist(x, u) then
                dist(x, u) = d
                record the shortest path from x to u that is through v
            end
            if d = dist(x, u) then
                record there is another shortest path from x to u that is through v
            end
        end
    end
end
Divide all BC(x) by 2
```

Definition 3 (Betweenness centrality) Given a graph $G = (X, E)$, the betweenness centrality of expert $x \in X$, denoted by $BC(x)$, is defined as $BC(x) = \sum_{x_s \neq x \neq x_t} \frac{\alpha_{st}(x)}{\alpha_{st}}$ where α_{st} is the total number of shortest paths between experts x_s and x_t and $\alpha_{st}(x)$ represents the number of those paths passing through x.

From Definition 3, expert x with high betweenness centrality indicates that x is on a large number of the shortest paths between the experts. In other words, an expert with high betweenness centrality means that this expert has shorter distances to more experts than an expert with lower betweenness centrality. As such, an expert with high betweenness centrality is more likely to have a lower leader distance. To calculate the betweenness centrality of each expert as well as the all-pair shortest path information, we introduce the BCinDijkstra algorithm derived from modifying the Dijkstra's algorithm. It is noted that, since the calculation of all-pair shortest paths between experts is necessary for team formation [11], the BCinDijkstra algorithm additionally incurs only a slight cost.

The BCinDijkstra algorithm is described in Algorithm 1. Initially, betweenness centrality BC is set to 0 for all experts. Meanwhile, the initial shortest distance between any two experts is set to infinity. For each expert $x \in X$, run Dijkstra's algorithm and build the shortest path tree $SPT(x)$ rooted at v. During the construction of $SPT(x)$, when a new expert x_n is added in the $SPT(x)$, increase the betweenness centrality of the experts on the paths from x_n to x by the fraction of the shortest paths that pass through the expert. If the cost of the path from x to u through v is less than

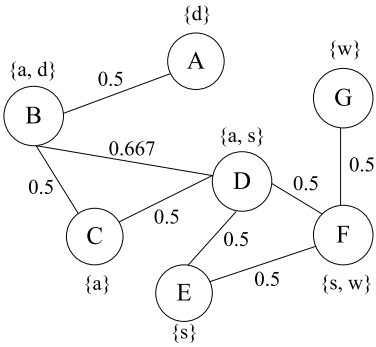


Fig. 2 A social network of experts

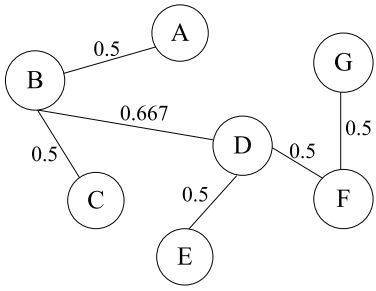


Fig. 3 The minimal spanning tree rooted at node A

the temporary cost of the path from x to u, record that the shortest path from x to u passes through v. If the costs are equal, it means that there is another shortest path from x to u with the same distance. In such a case, record the found path. The temporary shortest paths are all the shortest paths from x to u. At last, since all shortest paths will be computed twice, $BC(x)$ of each expert x is divided by 2.

We use Figs. 2 and 3 to illustrate the BCinDijkstra algorithm. Figure 2 shows a social network of experts where each circle represents an expert and the letter(s) next to a circle indicate(s) the possessed skills of the corresponding expert. The number next to an edge is the communication cost between the two endpoint experts. Assume that expert A invokes the BCinDijkstra algorithm. According to the Dijkstra's algorithm, the experts are processed in the descending order of their distances to expert A and thus the processing sequence is $A \rightarrow B \rightarrow C \rightarrow D \rightarrow E \rightarrow F \rightarrow G$. Then, the shortest path tree rooted at node A is shown in Fig. 3. As shown in the BCinDijkstra algorithm, when a new node is added in a shortest path tree, the betweenness centrality of each expert on the path from the new expert to the corresponding root is increased. For example, when node C is added in the shortest path tree rooted at A, $BC(B)$ is increased by 1. Similarly, in case of the addition of expert G, $BC(F)$, $BC(D)$, and $BC(B)$ are incremented by 1. When all the experts have built their shortest path trees, we have $BC(X) = \{0, 5, 0, 9, 0, 5, 0\}$. From $BC(X)$, experts D, B, and F are on multiple shortest paths and thus will have a higher chance to have the smallest communication cost to the team members as a leader.

Algorithm 2: BCPruning algorithm for team formation

Input: A project $P = \{s_1, s_2, \ldots, s_p\}$
Output: *bestTeam*, *bestLeader*, *bestLeaderDistance*

```
1  bestLeaderDistance = ∞
2  bestTeam = ∅
3  bestLeader = ∅
4  Sort X in the descending order based on betweenness centrality
5  for each expert x ∈ X do
6      leader = x
7      leaderDistance = 0
8      team = ∅
9      for each skill s ∈ P do
10         select expert v ∈ T(s) with the smallest distance to x with respect to s
11         leaderDistance = leaderDistance + dist(x, v)
12         add v to team
13         if leaderDistance ≥ bestLeaderDistance then
14             x cannot be the desired leader, skip x
15         end
16     end
17     bestLeader = leader
18     bestLeaderDistance = leaderDistance
19     bestTeam = team
20 end
21 return bestTeam, bestLeader, bestLeaderDistance
```

4.1.2 BCPruning algorithm

With the betweenness centrality information of all experts, we propose the BCPruning algorithm to prioritize leader candidates based on the values of their betweenness centrality and use the currently best leader distance to attempt to prevent each candidate from exhaustively searching the closest expert for each skill. The BCPruning algorithm is shown in Algorithm 2. In the beginning, the experts are sorted in a descending order based on the values of their betweenness centrality. Then, for each expert $x \in X$, x serves as a candidate leader and starts to form his team. For each required skill $s \in P$, x searches the closest expert v to x and adds $dist(x, v)$ to the *leaderDistance* of x. If the *leaderDistance* is larger than *bestLeaderDistance*, x cannot be the answer leader and thus the team formation process of x can terminate, saving the search cost. If the team search formation process finishes and x finds a team of experts with the *leaderDistance* $<$ *bestLeaderDistance*, x becomes the *bestLeader* and the *leaderDistance* of x is set to *bestLeaderDistance*.

We continue the example in Fig. 2 to explain the BCPruning algorithm. Because of $BC(X) = \{0, 5, 0, 9, 0, 5, 0\}$, the processing sequence of the candidates is $D \rightarrow B \rightarrow F \rightarrow A \rightarrow C \rightarrow E \rightarrow G$. Assume that a project with the required skill set {"web programming","data mining","software engineering"} is given. First, expert D is considered as a leader candidate. The best team of expert D is {{F:0.5, web programming},{B:0.667, data mining},{D:0, software engineering}} and the leader distance of expert D is 1.167, which is then assigned to *bestLeaderDistance*. Next, consider expert B as the second candidate. With respect to the required skill "web programming", the closet expert to B is expert F. The communication cost

of experts B and F, 1.167, is the same as *bestLeaderDistance*. As such, expert B cannot be the answer leader and the search process for the other skill "data mining" and "software engineering" can be terminated. Similarly, the early termination can be achieved for expert F searching team members having "data mining", expert A "web programming", expert C "data mining", and so on. Finally, expert D is the desired leader with team members $\{B, F\}$ for the given project. This example clearly shows that betweenness centrality is effective in selecting better initial candidates with lower leader distances and then helps to save the cost of searching the desired team and leader.

4.2 Skill spreading: SSPruning algorithm

Besides preferring the experts with high betweenness centrality as initial leader candidates, we devise an alternative algorithm, called the SSPruning algorithm. The main idea of the SSPruning algorithm is that each expert spreads his possessed skills to neighboring experts with distances less than the corresponding *skill distance*. A skill distance is used to enable an expert realize the distance lower bound between itself and the closest expert having the corresponding skill. With the spread of possessed skills of all the experts, an expert is able to compute a lower bound on its leader distance with respect to a given project. Similar to the BCPruning algorithm, the SSPruning algorithm can exploit the lower bound information to select initial promising candidates and to prune some experts as the answer leader, thereby reducing the computational cost.

4.2.1 Skill spreading

Before introducing the SSPruning algorithm, we explain how the skills possessed by each expert are spread to be known by neighboring experts. To reduce the computation cost of skill spreading, we propose the *skill distance* concept to determine how far a skill should be broadcast from an expert to the neighboring experts.

Definition 4 (Skill distance) Let $G' = (X', E')$ be a subgraph generated from random sampling of the graph $G = (X, E)$ of the social network of experts. Let $Diameter(G')$ and $Radius(G')$ denote the diameter and the radius of G', respectively. The skill distance SD_j of skill s_j is defined as $\frac{Diameter(G')+Radius(G')/2}{|T(s_j)|}$ where $|T(s_j)|$ is the cardinality of the set of the experts having skill s_j.

Employing the average of the diameter and the radius as the basis for the skill distance *SD* avoids insufficient skill information (*SD* is less than the radius) and high computational cost (*SD* is same as the diameter). Because a skill is usually possessed by multiple experts, the average is further divided by the number of experts with the skill in order to reduce the redundant information from skill spreading. Note that we use a subgraph for skill distance computation for a reduction in computational cost. To spread skills owned by each experts at a low cost, similar to the BCinDijkstra algorithm, we introduce the SSinDijkstra algorithm incorporating skill spreading with all-pair shortest path calculation.

Algorithm 3: SSinDijkstra algorithm

```
   Input: social network: G = (X, E)
   Output: all-pair shortest distances dist(s, t)∀s, t ∈ X and the received skill set IS(v)
1  Sample few vertices to create G' = (X', E'), run Dijkstra's algorithm, and compute the diameter
   and radius of G'
2  For each s_j, set the skill distance SD_j = ((diameter+radius)/2) / |X(s_j)|
3  Initialize dist(s, t)∀s, t ∈ X as ∞
4  for all expert x ∈ X do
5      Set spread skill set K to S(x)
6      Set dist(v, v) to 0
7      Set unvisited expert set UX = X
8      while UX is not empty do
9          Select the expert v with the smallest distance dist(x, v) from UX and remove v from UX;
10         if dist(x, v) is ∞ then
11             break
12         end
13         if dist(x, v) > SD_j of skill s_j in K then
14             remove s from K
15         end
16         if K is not empty then
17             spread K to u
18         else
19             stop spreading
20         end
21         for each neighbor u of v do
22             d = dist(x, v) + dist(v, u)
23             if d < dist(x, u) then
24                 dist(x, u) = d
25             end
26         end
27     end
28 end
```

Algorithm 3 shows the pseudo-code of the SSinDijkstra algorithm. To determine the skill distance, the diameter and the radius of a subgraph $G' = (X', E')$ created by random sampling of G are first computed with the Dijkstra's algorithm in line 1. With the diameter and the radius, the skill distance of each skill s_j can be set to the result of dividing the average of the diameter and the radius by $|T(s_j)|$ in line 2. Then, each expert spreads the skills possessed by him to only the neighboring experts within the corresponding skill distances in lines 4–28. Note that each expert maintains a received skill set that consists of skills spread by neighboring experts.

Again, we use Fig. 2 to introduce the SSinDijkstra algorithm. In the beginning, experts A and D are randomly selected to determine the diameter as 2.167 and the radius as 1.167, leading the skill distance of 1.667. Then, since experts A and B have the skill "data mining", the skill distance of data mining is set to 0.833 (1.667/2). With the SSinDijkstra algorithm, expert A spreads his skill "data mining" to the neighboring experts B and C because the distances between experts B and C and expert A are less than the skill distance of "data mining". Similarly, expert B has skills "algorithm" and "data mining" and the corresponding skill distances are 0.555 and 0.833, respectively. Because the distances from expert B to experts A and C are 0.5, expert

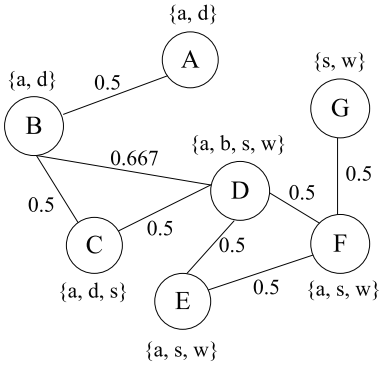


Fig. 4 An example of skill spreading of the experts

B spreads his skills "algorithm" and "data mining" to inform experts A and C. However, because the distance between experts B and D is 0.667, larger than 0.555, expert D will only know that expert B has the skill "data mining." For the other experts, the skills of expert B will not be known since they are too far away from expert B. The final result of skill spreading for the experts is shown in Fig. 4.

4.2.2 SSPruning algorithm

After the skill spreading process, each expert maintains the received skill set from neighboring experts together with the corresponding skill distances. With such information, an expert can compute a lower bound *SSLowerBound* of its leader distance when a project with a required skill set is given. When the lower bound *SSLowerBound* is larger than the current *bestLeaderDistance*, a leader candidate can be safely pruned. Besides, in case of *SSLowerBound* < *bestLeaderDistance*, a candidate searches team members with the required skills not contributing to *SSLowerBound*. The rationale is that the sum of the distances to those team members and *SSLowerBound* may be larger than *bestLeaderDistance*. If so, the search for experts with the skills contributing to *SSLowerBound* can be avoided. The SSPruning algorithm is shown in Algorithm 4. To start with, each expert computes his *SSLowerBound* with respect to the given project P in line 1. Select the experts as candidates in the descending order of the size of the received skill set in line 5. Then, each expert assigns *SSLowerBound* to *leaderDistance* in line 7. If *leaderDistance* > *bestLeaderDistance*, the current leader candidate can be safely pruned in lines 9–10. If a required skill exists in the received skill set, it is more likely for a candidate to find a close expert for such a skill. Thus, to increase the pruning possibility, each candidate searches the closest experts first for those skills that are in the received skill set and are required by the project. With the pruning idea, early formation termination can be achieved and thus save the total computational cost.

Here we give an example of the SSPruning algorithm. Assume that the skill distances of the skills are {algorithm (a): 0.555, data mining (d): 0.833, software engineering (s): 0.555, web programming (w): 0.833} and a given project requires the skill set {"web programming", "data mining", "software engineering"}. Consider

Algorithm 4: SSPruning algorithm for team formation

Input: A project $P = \{s_1, s_2, \ldots, s_p\}$
Output: *bestTeam*, *bestLeader*, *bestLeaderDistance*

compute lower bound *SSLowerBound* for all experts with respect to project P
bestLeaderDistance $= \infty$
bestTeam $= \emptyset$
bestLeader $= \emptyset$
for *each expert* $x \in X$ *in the descending order of the size of the received skill set* **do**
    leader $= x$
    leaderDistance $=$ *SSLowerBound*
    team $= \emptyset$
    if *leaderDistance* $>$ *bestLeaderDistance* **then**
        x cannot be the best leader, skip x
    end
    for *each skill* $s \in P \cap IS(x)$ *first, then* $s \in P \cap \neg IS(x)$ **do**
        select expert $v \in T(s)$ with the smallest distance to x
        leaderDistance $=$ *leaderDistance* $+$ *dist*(x, v)
        add v to *team*
        if *leaderDistance* $\geq$ *bestLeaderDistance* **then**
            x cannot be the best leader, skip x
        end
    end
    bestLeader $=$ *leader*
    bestLeaderDistance $=$ *leaderDistance*
    bestTeam $=$ *team*
end
return *bestTeam*, *bestLeader*, *bestLeaderDistance*

that expert A maintains the received skill set $\{a, d\}$, as shown in Fig. 4. With respect to "web programming", the distance between expert A and the closest expert with "web programming" must be larger than the corresponding skill distance 0.833 since "web programming" is not in the received skill set. A similar case is for expert A regarding "software engineering". Thus, for the given project, expert A has a lower bound on the leader distance of 1.388 $(0.833 + 0.5)$. Similarly, the lower bounds of the leader distances for experts B, C, D, E, F, and G are 1.388, 0.833, 0, 0.833, 0.833, and 0.833, respectively. Among the experts, expert D has the lowest leader distance bound 0 and thus serves as the first leader candidate. The leader distance of expert D is 1.167. With the leader distance of expert D being 1.167, experts A and B can be safely pruned since their lower bounds are larger than 1.167, saving the computation of team formation for experts A and B. On the other hand, the remaining experts C, E, F, and G cannot be pruned since their lower bounds are smaller than 1.167. Next, expert C is selected to be the leader candidate. Since expert C has the skill distance 0.833 for "web programming", expert C first searches the team members with the other three required skills. For "data mining", the closet expert to expert C is expert B and the distance between experts C and B is 0.5. Because *bestLeaderDistance* $= 1.167$ is smaller than 1.333 $(0.833 + 0.5)$, it is unlikely for expert C to become the answer leader and thus the team formation process terminates for expert C. By doing so, the answer leader is expert D for the given project.

5 Performance evaluation

In this section, we conduct experiments to evaluate the performance of the proposed algorithms in comparison of the prior work [11]. All the algorithms are implemented in Java and the experiments are run on an Intel® Core™ i7 2.93 GHz desktop PC with 4 GB of RAM. For performance measurement, we use the DBLP dataset extracted from the DBLP bibliography server on November 7, 2011. Following [11, 12], the DBLP dataset contains only the papers published in the prestigious conferences in the areas of Database = {*SIGMOD*, *VLDB*, *ICDE*, *ICDT*, *EDBT*, *PODS*}, DataMing = {*KDD*, *WWW*, *SDM*, *PKDD*, *ICDM*}, Artificial Intelligence={*ICML*, *ECML*, *COLT*, *UAI*}, and Theory = {*SODA*, *FOCS*, *STOC*, *STACS*}. An author is counted as an expert if the author has at least three papers in the extracted DBLP dataset. The skills of an expert is the set of terms appearing in the titles of at least two publications of the expert. The graph produced from the DBLP dataset consists of 8,154 experts and 3,804 edges. Two experts are connected in the graph if they have coauthored more than one paper. Note that the weight of the edge between two experts x_i and x_j is defined as $1 - \frac{p_{x_i} \cap p_{x_j}}{p_{x_i} \cup p_{x_j}}$ where p_{x_i} is the set of papers of author x_i. The number of required skills for each project is varied from 2 to 10.

To evaluate the performance, below we employ two performance metrics, namely the query execution time and the pruning ratio:

- Query execution time. The query execution time is defined as the CPU time of finding the best leader and the corresponding team.
- Pruning ratio. Let n and n_p respectively represent the total number of leader candidates and the number of the leader candidates that benefit from pruning of the proposed algorithm. The pruning ratio is defined as $\frac{n_p}{n}$.

Note that we do not list the leader distance since the proposed algorithms and the prior work all discover the team with the minimal leader distance. The experimental results are labeled "Kargar."

5.1 Impact of the number of required skills

In this experiment, we study the impact of the number of required skills for a project on the performance of the proposed algorithms and the prior work. The number of required skills increases from 2 to 10 in increments of 2. Figure 5(a) shows that the query execution time of all the algorithms increases as the number of required skills increases. This result agrees with the intuition that each leader candidate has to search more experts for team formation. We can observe from Fig. 5(a) that the proposed BCPruning and SSPruning algorithms substantially outperform the prior work because using betweenness centrality for initial candidate selection and spreading skills of experts are effective in reducing the search space of team members for candidates. Besides, as the number of required skills increases, the performance gains of the BCPruning and SSPruning algorithms get larger. Specifically, with the number of required skills increasing from 2 to 10, the BCPruning and SSPruning algorithms improve the query execution time from 29 % to 40 % and from 62 % to 64 %, respectively. That is, the BCPruning and SSPruning algorithms are respectively 1.42–1.68

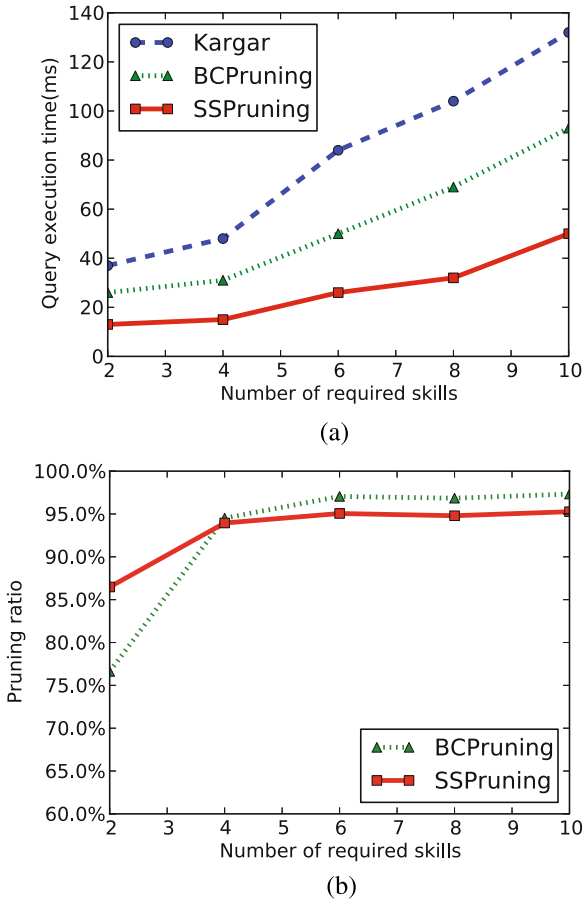
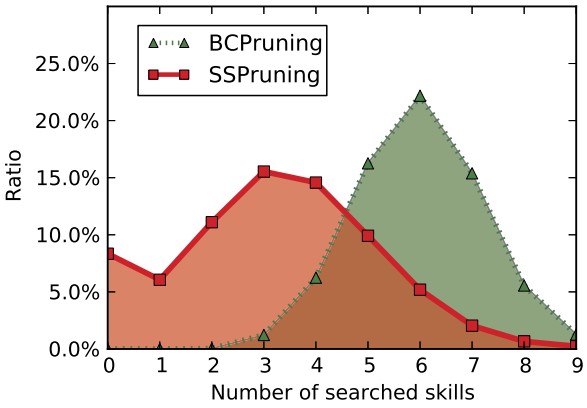


Fig. 5 Impact of the number of required skills: (**a**) query execution time; (**b**) pruning ratio

and 2.64–3.25 times faster than the prior work. The reason is that the pruning benefit of the proposed algorithms is more substantial for the larger number of required skills.

From Fig. 5(b), it can be seen that the BCPruning and SSPruning algorithms achieve high pruning ratios for all numbers of required skills. When the number of required skills is larger than 2, the pruning ratios of both of the proposed algorithms are higher than 93 %. This result indicates that the better initial candidate selection based on betweenness centrality in the BCPruning algorithm and the lower bounds enabled by skill spreading in the SSPruning algorithm are very effective in reducing the search space of team members for candidates. To provide a more fine-grained understanding of pruning benefits of the BCPruning and SSPruning algorithms, we present a breakdown of the numbers of searched skills on team formation termination for leader candidates in Fig. 6. Note that the number of required skills for a project is fixed at 10. As shown in Fig. 6, the SSPruning algorithm enables about 8.3 % of leader candidates to be directly pruned based on their lower bounds without searching for any team member. The vast majority of candidates could be excluded after searching for team members for fewer than 6 skills. On the other hand, the BCPrun-

Fig. 6 Breakdown



ing algorithm makes almost all the candidates terminate team formation before not more than 7 required skills are tackled. From Fig. 6, we can find that the SSPruning algorithm achieves earlier termination of team formation for leader candidates than the BCPruning algorithm, explaining the shorter query execution time of the SSPruning algorithm compared to the BCPruning algorithm.

5.2 Impact of the number of experts

We investigate the scalability of the proposed algorithms and the prior work by varying the number of experts in this experiment. To change the number of experts, we use the different thresholds for the number of papers that makes an author qualify as an expert, referring to [11]. With the threshold ranging from 2 to 6, the numbers of experts are in the range of 4k to 12k. The number of required skills for a project is set to 4. We observe from Fig. 7(a) that all the algorithms suffer from longer query execution time with the number of experts increasing. This is because the increase in the number of experts leads to more leader candidates and more experts for each required skill. In addition, as the number of experts becomes larger, the BCPruning and SSPruning algorithms are more effective in query execution time reduction since the pruning benefits are greater. Finally, Fig. 7(b) unveils that the pruning ratios of the BCPruning and SSPruning algorithms stay high regardless of the number of experts. This result verifies the scalability of the BCPruning and SSPruning algorithms.

6 Conclusions and future work

In this paper, we have proposed two efficient algorithms, namely BCPruning and SSPruning algorithms, for the problem of finding a team with a leader that has the required skill set and minimal communication cost. The BCPruning algorithm initially selects promising leader candidates that are on multiple shortest paths between the experts to obtain a lower communication cost earlier, achieving early termination of team member search for candidates. On the other hand, the SSPruning algorithm

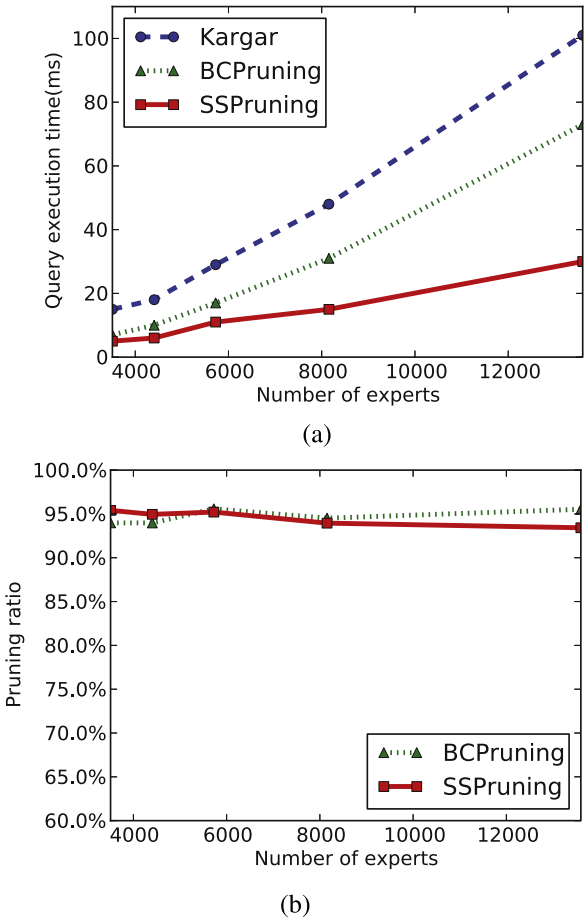


Fig. 7 Impact of the number of experts: (**a**) query execution time; (**b**) pruning ratio

requires each expert to spread its possessing skills to neighboring experts within the corresponding skill distance. The limited skill spreading benefits the lower bound determination, leader candidate selection, and team member search, thereby accelerating the discovery of the best leader and the corresponding team. The experimental results on the DBLP dataset show that the proposed BCPruning and SSPruning algorithms effectively reduce the query execution time by at most 40 % and 64 %, respectively. Besides, the results also indicate the scalability of the proposed algorithms compared with the prior work. In a future work, we will design new algorithms for the team formation problem without a leader. In addition, we plan to consider the load balancing and the degrees of skill mastery of experts for team formation.

Acknowledgements The authors thank the Taiwan Ministry of Economic Affairs and Institute for Information Industry for financially supporting this research (Industry and Government Application Empirical Environment Development Project).

References

1. Anagnostopoulos A, Becchetti L, Castillo C, Gionis A, Leonardi S (2010) Power in unity: forming teams in large-scale community systems. In: Proceedings of the 19th ACM international conference on information and knowledge management, pp 599–608
2. Anagnostopoulos A, Becchetti L, Castillo C, Gionis A, Leonardi S (2012) Online team formation in social networks. In: Proceedings of the 21st international conference on world wide web, pp 839–848
3. Backstrom L, Huttenlocher D, Kleinberg J, Lan X (2006) Group formation in large social networks: membership, growth, and evolution. In: Proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining, pp 44–54
4. Baykasoglu A, Dereli T, Das S (2007) Project team selection using fuzzy optimization approach. Cybern Syst 38(2):155–185
5. Buter B, Dijkshoorn N, Modolo D, Nguyen Q, van Noort S, van de Poel B, Salah AA, Salah AAA (2011) Explorative visualization and analysis of a social network for arts: the case of deviantART. J Converg 2(1):87–94
6. Chen SJ, Lin L (2004) Modeling team member characteristics for the formation of a multifunctional team in concurrent engineering. IEEE Trans Eng Manag 51(2):111–124
7. Fitzpatrick EL, Askin RG (2005) Forming effective worker teams with multi-functional skill requirements. Comput Ind Eng 48(3):593–608
8. Freeman LC (1977) A set of measures of centrality based on betweenness. Sociometry 40(1):35–41
9. Gaston ME, Simmons J, desJardins M (2004) Adapting network structure for efficient team formation. In: Proceedings of the AAAI 2004 fall symposium on artificial multi-agent learning
10. Jackson MO (2008) Network formation. In: The new Palgrave dictionary of economics and the law
11. Kargar M, An A (2011) Discovering top-k teams of experts with/without a leader in social networks. In: Proceedings of the 20th ACM international conference on information and knowledge management, pp 985–994
12. Lappas T, Liu K, Terzi E (2009) Finding a team of experts in social networks. In: Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining, pp 467–476
13. Li CT, Shan MK (2010) Team formation for generalized tasks in expertise social networks. In: IEEE international conference on social computing
14. Majumder A, Datta S, Naidu K (2012) Capacitated team formation problem on social networks. In: Proceedings of the 18th ACM SIGKDD international conference on knowledge discovery and data mining, pp 1005–1013
15. Pan R, Xu G, Fu B, Dolog P, Wang Z, Leginus M (2012) Improving recommendations by the clustering of tag neighbours. J Converg 3(1):13–20
16. Shtykh R, Jin Q (2011) A human-centric integrated approach to web information search and sharing. Hum-Cent Comput Inf Sci 1(2):1–37
17. Teraoka T (2012) Organization and exploration of heterogeneous personal data collected in daily life. Hum-Cent Comput Inf Sci 2(1):1–15
18. Wi H, Oh S, Mun J, Jung M (2009) A team formation model based on knowledge and collaboration. Expert Syst Appl 36(5):9121–9134
19. Zakarian A, Kusiak A (1999) Forming teams: an analytical approach. IIE Trans 31(1):85–97