Zhou Y, Huang JB, Jia XL *et al.* On participation constrained team formation. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 32(1): 139–154 Jan. 2017. DOI 10.1007/s11390-017-1710-6

# **On Participation Constrained Team Formation**

Yu Zhou<sup>1,2</sup>, Student Member, CCF, Jian-Bin Huang<sup>2,\*</sup>, Senior Member, CCF, Member, ACM Xiao-Lin Jia<sup>3</sup>, Member, CCF, and He-Li Sun<sup>3</sup>, Member, CCF

<sup>1</sup>School of Computer Science and Technology, Xidian University, Xi'an 710071, China
<sup>2</sup>School of Software, Xidian University, Xi'an 710071, China
<sup>3</sup>Department of Computer Science and Technology, Xi'an Jiaotong University, Xi'an 710049, China

E-mail: peterjone85@hotmail.com; jbhuang@xidian.edu.cn; {xlinjia, hlsun}@mail.xjtu.edu.cn

Received January 27, 2016; revised September 21, 2016.

**Abstract** The task assignment on the Internet has been widely applied to many areas, e.g., online labor market, online paper review and social activity organization. In this paper, we are concerned with the task assignment problem related to the online labor market, termed as CLUSTERHIRE. We improve the definition of the CLUSTERHIRE problem, and propose an efficient and effective algorithm, entitled INFLUENCE. In addition, we place a participation constraint on CLUSTERHIRE. It constrains the load of each expert in order to keep all members from overworking. For the participation-constrained CLUSTERHIRE problem, we devise two algorithms, named PROJECTFIRST and ERA. The former generates a participation-constrained team by adding experts to an initial team, and the latter generates a participation-constrained team by removing the experts with the minimum influence from the universe of experts. The experimental evaluations indicate that 1) INFLUENCE performs better than the state-of-the-art algorithms in terms of effectiveness and time efficiency; 2) PROJECTFIRST performs better than ERA in terms of time efficiency, yet ERA performs better than PROJECTFIRST in terms of effectiveness.

Keywords task assignment, team formation, universal project-team map

# 1 Introduction

Due to the wide application of the task assignment problem on the Internet, the team formation problem and the social event organization problem attract many researchers' attention. They can be applied to the online labor market (e.g., Guru and Freelancer) and the social networking services (e.g., Meetup and Plancast) respectively. The CLUSTERHIRE problem as a variant of the team formation problem was proposed in [1]. Given a collection of projects and a group of experts, it aims at finding a team of experts within the specified budget to maximize the total profit made by this team. In practice, it can be employed to provide teamhiring service to the company customers. Nonetheless, the CLUSTERHIRE problem merely sets up a profitmaximizing team for a group of projects, but fails to specify a specific team for each project. This may cause three results. 1) Some experts simultaneously join too many projects, and therefore they may be exhausted; 2) some experts may apply their skills to too many projects in parallel; 3) some experts may hold multiple positions in the same project. Therefore, it is necessary for us to improve the definition of the CLUSTERHIRE problem so that each selected expert is assigned to a specific team.

The authors of [1] first proved the complexity of the CLUSTERHIRE problem via the SETCOVER problem [2]. Then, they proposed three heuristic algorithms, called EXPERTGREEDY, PROJECTGREEDY and CLIQUEGREEDY. However, these three algorithms take relatively simple strategies to select experts into the final team. This causes the profits made by them

Regular Paper

The work was partially supported by the National Natural Science Foundation of China under Grant Nos. 61472299, 61540008, 61672417 and 61602354, the Fundamental Research Funds for the Central Universities of China under Grant No. BDY10, the Shaanxi Postdoctoral Science Foundation, and the Natural Science Basic Research Plan of Shaanxi Province of China under Grant No. 2014JQ8359.

 $<sup>^{*}</sup>$ Corresponding Author

 $<sup>\</sup>textcircled{O}2017$ Springer Science + Business Media, LLC & Science Press, China

to be quite less than the optimal value. In this paper, we employ an expert-skill-project tripartite graph to describe the CLUSTERHIRE problem only for con-Then, we propose an efficient and effecvenience. tive algorithm, namely INFLUENCE, for forming a costconstrained profit-maximizing team. This approach takes as input a universal project-team map which can be viewed as a compressed version of the expert-skillproject tripartite graph. It can be obtained from the expert-skill-project tripartite graph. In the universal project-team map, the expert with the minimum influence is removed until the total cost of the remaining experts does not exceed the specified budget. Experimental evaluation suggests that INFLUENCE performs better than the state-of-the-art algorithms in terms of both the effectiveness and the time efficiency. Moreover, we discover that some experts in the team formed by the above method may participate in too many projects simultaneously. This causes them to be exhausted, because the amount of their work is beyond their capacity. Note in particular that Golshan et al. who proposed the CLUSTERHIRE problem also considered this scenario, and defined the T-CLUSTERHIRE problem in [1]. The problem constrains the number of projects for which an expert can utilize a skill. However, this constraint has the limitation of causing an expert to participate in too many projects simultaneously under some extreme scenarios. For example, an expert possesses 100 skills, which are required by 100 different projects respectively. Each project requires one single skill, and is covered by the team yielded by some algorithm. At this time, the expert will simultaneously join too many projects, and therefore be exhausted. To overcome the limitation, we place a participation constraint on the CLUSTERHIRE problem, which directly restricts the number of projects that each expert participates in. This is obviously distinct from the constraint in T-CLUSTERHIRE.

The participation-constrained CLUSTERHIRE problem is also intractable, since its simplified version is the CLUSTERHIRE problem. We devise two algorithms, namely PROJECTFIRST and ERA, for this problem. PROJECTFIRST spends less time on yielding a profit-maximizing feasible team than ERA, whereas ERA makes larger profit than PROJECTFIRST. Concretely, PROJECTFIRST takes an order-first strategy to merge sub-teams into the final team. Reversely, ERA yields a participation-constrained profit-maximizing team within the specified budget by successively removing the minimum influence experts from the universe of experts. Experimental evaluations suggest that 1) ERA performs better than PROJECTFIRST in terms of the effectiveness, yet PROJECTFIRST performs better than ERA in terms of the time efficiency; 2) it is necessary for us to propose the participation-constrained CLUSTER-HIRE problem, because load unbalance really occurs.

Contributions. Our main contributions are summarized as follows. 1) We improve the definition of the CLUSTERHIRE problem, and propose an effective and efficient algorithm for the improved CLUSTERHIRE problem. It not only produces a profit-maximizing team within the specified budget, but also can specify a team for each project. 2) We place a participation constraint on the improved CLUSTERHIRE problem in order to keep the experts from overworking. 3) We devise two algorithms, namely PROJECTFIRST and ERA, for forming a participation-constrained team, and evaluate their effectiveness and efficiency.

*Roadmap.* The rest of this paper is organized as follows. Section 2 introduces some work related to the task assignment problem and the team formation problem. Section 3 gives some notations, definitions and the problem statement. Section 4 gives a new approach to solve the CLUSTERHIRE problem. Section 5 provides two algorithms for the participation-constrained CLUS-TERHIRE problem. Section 6 depicts the experimental evaluation, and Section 7 concludes the paper.

# 2 Related Work

The team formation problem and the task assignment problem have been studied for years. Below, we provide an overview of the work related to them.

Socialized Team Formation. To the best of our knowledge, the team formation was first studied in [3]. This paper proposed a structured and unified methodology for the team formation problem. Lappas et al. explored the team formation problem on social networks in [4]. This paper studied the problem of finding a team of experts with the minimum communication cost to finish a given project. Since then, many researchers extended this work. Anagnostonpoulos et al. presented a general framework for the team formation problem on social networks in [5], and studied the online team formation problem on social networks in [6]. Kargar and An explored the problem of discovering the top-k teams of experts with/without a leader on social networks. Datta *et al.*<sup>[8]</sup> considered the capacities of experts and</sup> studied the problem of forming a capacitated team on social networks. Roy et al.<sup>[9]</sup> associated group recommendations with group formation. Li  $et \ al.^{[10]}$  proposed fast algorithms for team member recommendation.

Social Activity Organization. Li and Shan<sup>[11]</sup> studied the problem of automatically composing activity groups in a social network according to user-specified activity information. Li et al.<sup>[12]</sup> proposed the social event organization (SEO) problem. This problem assigns a collection of events for a group of users to attend, where users are embedded in a social network and have innate levels of interest in the events. Feng etal.<sup>[13]</sup> considered the problem of finding influential social organizers in online social networks. Shuai et al.<sup>[14]</sup> formulated a new problem, named Willingness mAximization for Social Group (WASO) and devised an efficient and effective randomized algorithm for it. She etal.<sup>[15-16]</sup> studied the conflict-aware event-participant arrangement problem and the utility-aware social eventparticipant planning problem. Shen et al.<sup>[17]</sup> studied the problem of maximizing the friend-making likelihood for social activity organizations. Armenatzoglou et al. developed a game-theoretic framework for the real-time multi-criteria social graph partitioning in [18]. However, their objective function only considers the connectivity and similarity among users. This is different from our objective function which considers the profit of projects and the cost of experts. Tong et al.<sup>[19]</sup> studied the problem of recommending users to events on the event-based social network under the bottleneck scenarios. They considered three influential recommendation strategies: spatial locations of events and users, attribute similarities between events and users, and friend relationships among users. We do not consider the aforementioned factors in the settings of our problem, because we only focus on the online collaboration of experts all over the world in the online labor market, e.g., Guru and Freelancer.

Reviewer Assignment Problem. To the best of our knowledge, Susan and Dumais first studied the automatic assignment of submitted manuscripts to reviewers in [20]. Karimzadehgan *et al.*<sup>[21]</sup> studied how to model multiple aspects of expertise and assign reviewers so that they together can cover all subtopics in the document well. Kou *et al.*<sup>[22]</sup> proposed a generalized framework for fair reviewer assignment. Mimno and McCallum<sup>[23]</sup> evaluated several methods for measuring the affinity of a reviewer to a paper.

# 3 Preliminaries

# 3.1 Notations

Let  $\mathcal{G} = (\mathcal{X}, \mathcal{S}, \mathcal{P}, C, \delta, F)$  be an expert-skill-project tripartite graph, where  $\mathcal{X}$  is a set of experts,  $\mathcal{S}$  is a set of skills, and  $\mathcal{P}$  is a set of projects. Note in particular that there are no edges between experts and projects.  $\forall e \in \mathcal{X}, C(e)$  denotes the cost of e when hiring it and  $\delta(e)$  denotes the capacity of e, which is the maximum number of projects that it can participate in.  $\forall p \in \mathcal{P}, F(p)$  denotes the profit of p upon completion.  $\forall v \in \mathcal{X} \cup \mathcal{S} \cup \mathcal{P}, N(v)$  denotes the neighborhood of v and d(v) denotes the degree of v. Obviously, d(v) = |N(v)|. Especially, N(e) represents the set of skills that e possesses for any  $e \in \mathcal{X}$ , and N(p) represents the set of skills that p requires for any  $p \in \mathcal{P}$ . For any  $s \in \mathcal{S}$ ,  $N_{\mathcal{X}}(s)$  represents the set of experts that possess s and  $N_{\mathcal{P}}(s)$  represents the set of projects that require s. Obviously,  $N(s) = N_{\mathcal{X}}(s) \cup N_{\mathcal{P}}(s)$ .

A subset T of  $\mathcal{X}$  is called a team. A team T can finish a project  $p \in \mathcal{P}$  only when the skill set N(p) of p is covered by the skill set N(T) of T, where N(T)denotes the set of skills that more than one expert in T possesses. In addition, C(T) denotes the total cost of T. Obviously,  $N(T) = \bigcup_{e \in T} N(e)$  and  $C(T) = \sum_{e \in T} C(e)$ .  $\forall P \subseteq \mathcal{P}, N(P)$  denotes the set of skills that more than one project in P requires and F(P) denotes the total profit of projects in P. Therefore,  $N(P) = \bigcup_{p \in P} N(p)$  and  $F(P) = \sum_{p \in P} F(p)$ .

#### 3.2 Problem Statement

Below, we define two problems on an expert-skillproject tripartite graph  $\mathcal{G}$ . They seek to find a set of projects  $P = \{p_{i_1}, p_{i_2}, \dots, p_{i_k}\}$  and a group of teams  $\{T_{i_1}, T_{i_2}, \dots, T_{i_k}\}$  such that  $N(p_{i_s}) \subseteq N(T_{i_s})$  for any  $i \in \{1, 2, \dots, k\}$ . Let  $T = \bigcup_{s=1}^k T_{i_s}$ .

IMPROVEDCH. For any  $e \in \mathcal{X}$ , the capacity of e is set to an extremely large number. For simplification, the expert-skill-project tripartite graph is denoted as  $\mathcal{G} = (\mathcal{X}, \mathcal{S}, \mathcal{P}, C, F)$ . The goal is to maximize the total profit F(P) of P under the cost constraint  $C(T) \leq B$ , where B is the specified budget.

PARTCONSCH. For any  $e \in \mathcal{X}$ , n(e) denotes the actual number of projects that e participates in. The goal is to maximize the total profit F(P) of P under the cost constraint  $C(T) \leq B$  and the participation constraint  $n(e) \leq \delta(e), \forall e \in T$ .

In form, the problem IMPROVEDCH is similar to the problem CLUSTERHIRE proposed in [1]. However, there still have some differences between them. CLUS-TERHIRE aims to find a team  $T \subseteq \mathcal{X}$  such that the total profit of projects covered by T is maximized and the total cost of T does not exceed B. Obviously, it only finds a team of experts and a group of projects covered by them, but does not specify a corresponding team for each project. IMPROVEDCH is capable of specifying a corresponding team for each project. In fact, the optimal solution of IMPROVEDCH is identical to that of CLUSTERHIRE.

**Lemma 1**. The problem IMPROVEDCH is equivalent to the problem CLUSTERHIRE.

*Proof.* First, we prove that any optimal solution of IMPROVEDCH is an optimal solution of CLUSTER-HIRE. Suppose that  $P = \{p_{i_1}, \cdots, p_{i_k}\}$  and  $\mathcal{T} =$  $\{T_{i_1}, \cdots, T_{i_k}\}$  such that  $N(p_{i_s}) \subseteq N(T_{i_s})$  for any  $s \in$  $\{1, 2, \cdots, k\}$  is any optimal solution of IMPROVEDCH. Below, we prove that  $T = \bigcup_{s=1}^{k} T_{i_s}$  is the optimal solution of CLUSTERHIRE. Obviously,  $N(P) \subseteq N(T)$ . Assume that T is not the optimal solution of CLUS-TERHIRE. Without loss of generality, let  $T' \neq T$  be an optimal solution of CLUSTERHIRE. Let P' be the set of projects covered by T'. Obviously, F(P') > F(P)and  $C(T') \leq B$ . For any  $p \in P'$ , let  $T'(p) = \{e \in$  $T'|N(e) \cap N(p) \neq \emptyset$ . We have  $N(p) \subseteq T'(p)$  because  $N(P') \subseteq N(T')$ . In addition,  $\bigcup_{p \in P'} T'(p) \subseteq T'$ . Therefore, P' and  $\{T(p)|p \in P'\}$  are a solution of IM-PROVEDCH and  $C(\bigcup_{p \in P'} T(p)) \leq C(T') \leq B$ . Obviously, this contradicts the fact that P and  $\mathcal{T}$  are the optimal solution of IMPROVEDCH. Therefore, any optimal solution of IMPROVEDCH is an optimal solution of ClusterHire.

Likewise we can prove that any solution of CLUS-TERHIRE is a solution of IMPROVEDCH.

**Lemma 2**. The decision version of IMPROVEDCH is NP-complete and NP-hard to approximate.

**Proof.** According to Lemma 1, IMPROVEDCH is equivalent to CLUSTERHIRE. In [1], it has been proved that the decision version of CLUSTERHIRE is NP-complete and NP-hard to approximate. Therefore, the decision version of IMPROVEDCH is NP-complete and NP-hard to approximate.  $\Box$ 

**Lemma 3**. The decision version of PARTCONSCH is NP-complete and NP-hard to approximate.

*Proof.* According to the definitions of problem IMPROVEDCH and problem PARTCONSCH, IM-PROVEDCH is a simplified version of PARTCONSCH. The decision version of IMPROVEDCH is NP-complete and NP-hard to approximate according to Lemma 2. Therefore, the decision version of PARTCONSCH is NPcomplete and NP-hard to approximate.  $\hfill \Box$ 

Before proceeding, we first introduce a useful data representation named a universal project-team map, which is actually a compressed version of the original expert-skill-project tripartite graph  $\mathcal{G}$ . For a project  $p \in \mathcal{P}$ , p is accomplishable in the context of problem IMPROVEDCH if we can find a team T to accomplish p, i.e.,  $N(p) \subseteq N(T)$ . In particular, p is accomplishable in the context of problem PARTCONSCH if we can find a team T satisfying the participation constraint to accomplish p, i.e.,  $N(p) \subseteq N(T)$ . A team satisfies the participation constraint, if the current workloads of its members are less than their capacities. Below, we give two preprocessing algorithms, named UPTMA for problem IMPROVEDCH and PREPROCESS for problem PARTCONSCH, to get the universal project-team map.

**Definition 1** (Universal Project-Team Map). An universal project-team map is a set of entries, each of which consists of an accomplishable project p and its corresponding team T such that  $N(p) \subseteq N(T)$ .

*Example* 1.  $\mathcal{G}_e = \{\mathcal{X}_e, \mathcal{S}_e, \mathcal{P}_e, C_e, \delta_e, F_e\}$  is an expert-skill-project tripartite graph, where  $\mathcal{X}_e = \{e_1, e_2, e_3\}, \mathcal{S}_e = \{s_1, s_2, s_3, s_4\}, \text{ and } \mathcal{P}_e = \{p_1, p_2, p_3\}.$  The structure of the expert-skill-project tripartite graph is shown in Fig.1. The budget is set to 9.



Fig.1. Structure of the tripartite graph in example 1.

### 4 ImprovedCH Problem

In this section, we introduce an efficient algorithm for the problem IMPROVEDCH. This algorithm takes a universal project-team map as its input. Therefore, we need to derive the universal project-team-map from the expert-skill-project tripartite graph  $\mathcal{G} = (\mathcal{X}, \mathcal{S}, \mathcal{P}, C, F)$ via a preprocessing procedure. The procedure is based on voting in project-skill-expert order, which is introduced in Subsection 4.1, and assigning each skill to a single expert, which is introduced in Subsection 4.2. It is worth noting that Subsection 4.2 needs to use the algorithm depicted in Subsection 4.1. After obtaining the universal project-team map via the algorithm depicted in Subsection 4.2, we compute the influence of each expert which is the ratio of the total profit of his/her influenced projects to his/her cost using Definition 7 in Subsection 4.3. The influenced projects of an expert are those that cannot be finished without this expert. Then, we remove the expert with the minimum influence until the total cost of remaining experts does not exceed B. This procedure is introduced in Subsection 4.3.

### 4.1 Voting

Before proceeding, we introduce some necessary definitions that will be used in the following. All of them are defined on an expert-skill-project tripartite graph  $\mathcal{G}$ . These definitions are used to assess the potential of experts to participate in some projects. The profits of projects are evenly distributed to the skills associated with them. Thereby, each skill obtains a score that is used to assess the value of this skill. Furthermore, the potential of each expert can be defined using the scores of the skills that he/she possesses and his/her cost.

**Definition 2** (Voting). For any  $p \in \mathcal{P}$ , p votes  $\frac{F(p)}{d(p)}$  for each skill in N(p). This process is called voting.

**Definition 3** (Vote). For any  $s \in S$ , s gets a vote  $\frac{F(p)}{d(p)}$  from each  $p \in N_{\mathcal{P}}(s)$ .  $\mathcal{V}(s)$  is a map consisting of  $p \in N_{\mathcal{P}}(s)$  and the vote  $\frac{F(p)}{d(p)}$ . Therefore,  $\mathcal{V}(s) = \{(p, \frac{F(p)}{d(p)}) | p \in N_{\mathcal{P}}(s)\}.$ 

**Definition** 4 (Score). For any  $s \in S$ , Score(s) is the sum of votes that s gets. That is,

$$Score(s) = \sum_{\substack{(p, \frac{F(p)}{d(p)}) \in \mathcal{V}(s)}} \frac{F(p)}{d(p)}.$$
 (1)

**Definition 5** (Potential). For any  $e \in \mathcal{X}$ , Potential(e) is the ratio of the total score of  $s \in N(e)$ to the cost of e, i.e.,

$$Potential(e) = \frac{\sum_{s \in N(e)} Score(s)}{C(e)}.$$
 (2)

Now, we introduce a method of assessing the potentials of experts to participate in some projects. This method is based on voting in project-skill-expert order. Concretely, each skill s gets votes  $\mathcal{V}(s)$  from the projects associated with it, and thereby is associated with a score score(s). Each expert gets all these scores of the skills associated with him/her, and thereby we can compute his/her potential of participating in some projects.

For any  $p \in \mathcal{P}$ , p requires all of the skills in N(p). That means the profit F(p) of p can be partitioned into d(p) pieces, where d(p) = |N(p)|. Each skill in N(p) gets  $\frac{F(p)}{d(p)}$ . In other words, each project  $p \in \mathcal{P}$  votes  $\frac{F(p)}{d(p)}$  for skills in N(p).

For each skill  $s \in S$ , s gets a vote  $\frac{F(p)}{d(p)}$  from each  $p \in N_{\mathcal{P}}(s)$ . All votes that s gets are denoted as  $\mathcal{V}(s)$  according to Definition 3. According to Definition 4, each skill  $s \in S$  is associated with a score Score(s). The larger Score(s) is, the more important s is.

For each expert  $e \in \mathcal{X}$ , e possesses the skills in N(e). This means e gets a vote Score(s) from  $s \in N(e)$ . According to Definition 5, e is associated with Potential(e). The larger Potential(e) is, the larger the potential of e is.

Above all, we get the potential of e for any  $e \in \mathcal{X}$ . Algorithm 1, termed as VOTING, describes the above procedure. After voting in project-skill-expert order, each skill s is associated with a vote set  $\mathcal{V}(s)$ . Below, we give a new problem on the expert-skill-project tripartite graph  $\mathcal{G}$ , in which each skill is associated with a vote set  $\mathcal{V}(s)$ . For any two different skills  $s_1$  and  $s_2$ ,  $\mathcal{V}(s_1) + \mathcal{V}(s_2)$  is defined as  $\mathcal{V}(s_1) + \mathcal{V}(s_2) = \{(p, \frac{F(p)}{d(p)}) | p \in \mathcal{V}(s_1) - \mathcal{V}(s_2)\} \bigcup \{(p, \frac{2 \times F(p)}{d(p)}) | p \in \mathcal{V}(s_1) - \mathcal{V}(s_2)\} \bigcup \{(p, \frac{F(p)}{d(p)}) | p \in \mathcal{V}(s_2) - \mathcal{V}(s_1)\}$ . Let S be a subset of  $\mathcal{S}$ .  $\forall (p, x) \in \sum_{s \in S} \mathcal{V}(s)$ , where  $x = \frac{y \times F(p)}{d(p)}$ ,  $\exists y \leq |S|$ . p is called a complete project in S if F(p) = x.

Alg	gorit	hm	1.	V	OTIN	G
-----	-------	----	----	---	------	---

INPUT: expert-skill-project tripartite graph  $\mathcal{G} = (\mathcal{X}, \mathcal{S}, \mathcal{P}, C, F),$ where  $n = |\mathcal{X}|, m = |\mathcal{S}|$  and  $l = |\mathcal{P}|$ **OUTPUT:** a potential set 1 potentialSet  $\leftarrow \emptyset$ ; 2 for each s in S do  $\mathcal{V}(s) \leftarrow \emptyset;$ 3 for each p in  $N_{\mathcal{P}}(s)$  do 4  $\mathcal{V}(s) \leftarrow \mathcal{V}(s) \cup \{(p, \frac{F(p)}{d(p)})\};$ 5Compute Score(s) according to (1); 7 for each e in  ${\mathcal X}$  do Compute Potential(e) according to (2); 8 9  $potentialSet \leftarrow potentialSet \cup \{Potential(e)\};$ 10 Return *potentialSet*;

Problem 1.  $\mathcal{G}$  is an expert-skill-project tripartite graph, in which each skill is associated with  $\mathcal{V}(s)$ . B is a specified budget. Our goal is to find a team T within the budget B such that the total profit of the complete projects in  $\bigcup_{e \in T} N(e)$  is maximized. **Lemma 4.** The optimal solution of problem IM-PROVEDCH is identical to that of problem 1.

*Proof.* We first prove that any optimal solution of problem IMPROVEDCH is an optimal solution of problem 1. Let  $\{p_{i_1}, \dots, p_{i_k}\}$  be a project set and  $\{T_{i_1}, \cdots, T_{i_k}\}$  be a group of teams such that  $T_{i_s}$  covers  $p_{i_s}, s = 1, 2, \cdots, k.$   $T = \bigcup_{s=1}^k T_{i_s}$  is an optimal solution of problem IMPROVEDCH. Obviously,  $C(T) \leq B$ . Now, we prove T is an optimal solution of problem 1 by contradiction. Assume that T is not the optimal solution of problem 1. Without loss of generality, let  $T' \neq T$  be an optimal solution of problem 1. Therefore, the total profit of the complete projects in  $\bigcup_{e \in T'} N(e)$  is larger than that of the complete projects in  $\bigcup_{e \in T} N(e)$ . In addition,  $C(T') \leq B$ . Therefore, T' is an feasible solution of problem IMPROVEDCH. Obviously, this contradicts the fact that T is the optimal solution of problem IM-PROVEDCH.

Likewise, we can prove any optimal solution of problem 1 is also the optimal solution of problem IM-PROVEDCH.  $\hfill \Box$ 

According to Lemma 4, we only need to solve problem 1. Therefore, we need to find a team  $T \subseteq \mathcal{X}$  such that the total profit of the complete projects in  $\bigcup_{e \in T} N(e)$  is maximized. For this purpose, it is a reasonable strategy to first get a universal project-team map from the expert-skill-project tripartite graph and then successively remove the expert with the minimum influence until the total cost of the remaining experts does not exceed the budget.

#### 4.2 Mapping from Skills to Experts

Below, we introduce a preprocessing algorithm (Algorithm 2), termed as UPTMA, for generating a universal project-team map. It takes as input an expert-skillproject tripartite graph and invokes algorithm VOTING to compute the potentials of experts. Then, the expert with the maximum potential value is selected to undertake the skills associated with him/her. These covered skills are removed from the expert-skill-project tripartite graph. Repeat the above process until all skills are covered. According to Definition 1, the universal project-team map is made up of an accomplishable project and its corresponding team. Note that problem IMPROVEDCH does not consider the capacity of each expert. Therefore, each project in  $\mathcal{P}$  can be potentially accomplished.

Let *expertSkillMap* be a map that associates an expert with a set of skills. Let *remainingSkillSet* 

be the set of skills that has not been selected. Initially, remainingSkillSet is set to S. potentialSet is yielded by VOTING. The entry consisting of the expert e with the maximum potential and N(e) is added to expertSkillMap. The skills in N(e) are removed from remainingSkillSet. e and the edges adjacent to it are removed from  $\mathcal{G}$ . Below, we repeat the following procedure until remainingSkillSet is empty. We invoke VOTING on  $\mathcal{G}$  and get a new potentialSet. Suppose  $e_{\text{new}}$  is the current expert with the maximum potential. Let  $value = N(e_{\text{new}}) \cap remainingSkillSet$ . The entry  $(e_{\text{new}}, value)$  is added to expertSkillMap. The skills in value are removed from remainingSkillSet. In addition,  $e_{\text{new}}$  and the edges adjacent to it are removed from  $\mathcal{G}$ .

#### Algorithm 2. UPTMA

Input: expert-skill-project tripartite graph  $\mathcal{G} = (\mathcal{X}, \mathcal{S}, \mathcal{P}, C, F)$ , where  $n = |\mathcal{X}|, m = |\mathcal{S}|$  and  $l = |\mathcal{P}|$ Output: a universal project-team map 1 Initialize two empty maps expertSkillMap and  $projectTeamMap; remainingSkillSet \leftarrow S; G' \leftarrow G;$ 2 while  $remainingSkillSet \neq \emptyset$  do  $potentialSet \leftarrow VOTING(\mathcal{G}');$ 3 4  $e \leftarrow \arg \max_{potential(e') \in potentialSet} potential(e');$ 5 value  $\leftarrow N(e) \cap remainingSkillSet;$ 6 Add (e, value) to expertSkillMap; 7  $remainingSkillSet \leftarrow remainingSkillSet - value;$ 8 Remove e and the edges adjacent to e from  $\mathcal{G}'$ ; 9 for each p in  $\mathcal{P}$  do 10  $pTeam \leftarrow \emptyset;$ for each e in expertSkillMap do 11 12if  $N(p) \cap expertSkillMap(e) \neq \emptyset$  do  $pTeam \leftarrow pTeam \cup \{e\};$ 1314 projectTeamMap(p) = pTeam;15 Return projectTeamMap;

As a result, we get a universal project-team map, namely projectTeamMap, where the keys are the projects in  $\mathcal{P}$  and the values are generated by the map expertSkillMap. The algorithm UPTMA describes the above procedure.

# 4.3 Forming an Improved Team

UPTMA, as a preprocessing procedure, yields a universal project-team map U. Without loss of generality, let  $U = \{(p_1, T_1), \dots, (p_s, T_s)\}$ , and  $T = \bigcup_{i=1}^{s} T_i$ . Below, we first give some useful definitions on U. These definitions are used to assess the influence of each expert, i.e., the price paid for removing it.

**Definition 6** (Dominated Project Set). For any  $e \in T$ ,  $\mathcal{D}(e)$  is the set of projects dominated by e. That is to say, all projects in  $\mathcal{D}(e)$  cannot be finished when e is absent from T. Therefore,  $\mathcal{D}(e) = \{p \in U | e \in U(p)\}$ .

Yu Zhou et al.: On Participation Constrained Team Formation

**Definition 7** (Influence). For any  $e \in T_U$ , Influence(e) is the ratio of the total profit of projects dominated by e to the cost of e. That is,

$$Influence(e) = \frac{\sum_{p \in \mathcal{D}(e)} F(p)}{C(e)}.$$
 (3)

According to Definition 7, the influence of experts in T is used to evaluate the relative importance of experts. The larger the influence is, the larger the possibility that e is left is. Therefore, we successively remove the expert with the minimum influence until the total cost of remaining experts does not exceed B. At last, we obtain a cost-constrained map that associates a project with its corresponding team. The algorithm INFLU-ENCE (Algorithm 3) describes the above procedure.

In Algorithm 3 INFLUENCE, lines 2~3 take O(|T|) to compute the influence of each expert. Lines 5~8 spend at most  $O(|T|^2)$  on removing useless experts and relevant projects. Lines 9~12 take at most O(|U|) to form a map whose entries consist of projects and corresponding team. Therefore, the worst-case time complexity of INFLUENCE is  $O(|T|^2 + |U|)$ . For INFLUENCE, we only need to store the project-team entries in the project-team map U. Therefore, the worst-case space complexity of INFLUENCE is  $O(|U| + \sum_{i=1}^{n} |T_i|)$ .

Algorithm 3. INFLUENCE

Input: universal project-team map U, budget BOutput: a cost-constrained project-team map  $1 T = \bigcup_{i=1}^{n} T_i;$ 2 for each e in T do Compute Influence(e) according to (3); 3 4 cost  $\leftarrow C(T)$ : 5 while cost > B do  $e^* = \arg \min_{e \in T} Influence(e);$  $T \leftarrow T - \{e\};$ 7  $cost \leftarrow C(T);$ 8 9 for each p in U do  $tempT \leftarrow U(p);$ 10 11 if  $tempT \not\subseteq T$  do Remove (p, tempT) from U; 1213 Return U;

Below, we exemplify INFLUENCE with example 1. UPTMA, as a preprocessing procedure, is used to derive a universal project-team map. It first assigns each skill to a single expert.  $s_1$ ,  $s_2$  and  $s_3$  are assigned to  $e_1$ , and  $s_4$  is assigned to  $e_3$ . Thus, the universal project-team map is  $U = \{(p_1, \{e_1\}), (p_2, \{e_1, e_3\}), (p_3, \{e_1, e_3\})\}$ . Then, the influence of each expert is computed according to (3). The expert with the minimum influence is removed until the total cost of remaining experts does not exceed B. In this example, U is the final projectteam map, because  $C(e_1) + C(e_3) \leq B$ . The complete process is shown in Fig.2.



Fig.2. Illustration of algorithm INFLUENCE.

# 146

#### J. Comput. Sci. & Technol., Jan. 2017, Vol.32, No.1

## 5 PartConsCH Problem

The IMPROVEDCH problem does not consider the workloads of experts. This gives rise to load unbalance, and therefore is unfair for some experts with larger workload. On account of the energy of each expert and the fairness, we should constrain the number of projects that each expert can participate in. This scenario is modeled by problem PARTCONSCH.

According to Lemma 3, the decision version of PARTCONSCH is NP-complete and NP-hard to approximate. We propose two algorithms, termed as PROJECTFIRST and ERA, to form a participationconstrained profit-maximizing team within the specified budget. They all use the greedy algorithm for the SETCOVER problem which is introduced in [2], termed GREEDYSETCOVER.

# 5.1 PROJECTFIRST Algorithm

This Subsection introduces an effective algorithm (Algorithm 4), named PROJECTFIRST, to form a participation-constrained profit-maximizing team within the specified budget. We first initialize a map projectTeamMap that associates a project with its corresponding team. For each project  $p \in \mathcal{P}$ , the nonredundant team T that covers p is computed by GREEDYSETCOVER. Let  $ratio(p) = \frac{F(p)}{C(T)}$  for any project p. We select the project  $p^*$  and its corresponding team T<sup>\*</sup> such that  $\frac{F(p^*)}{C(T^*)} = \max_{p \in \mathcal{P}} \frac{F(p)}{C(p)}$ . The entry consisting of  $p^*$  and T<sup>\*</sup> is added to projectTeamMap. The capacities of experts in T<sup>\*</sup> decrease by 1.  $p^*$  is removed from  $\mathcal{P}$ .  $(p^*, T^*)$  is viewed as a seed to merge with other project-team tuples.

For each project  $p \in \mathcal{P}$ , the expert e such that c(e) > 0 and  $N(e) \cap N(p) \neq \emptyset$  is added to the candidate expert set *pce* of p. The irredundant team T that covers p is computed by GREEDYSETCOVER. If T = None, p cannot be accomplished. Otherwise, let  $ratio(p) = \frac{F(p)}{C(T)}$  for any available project p. At this time, we take an order-first selection strategy which is to select the first project-team tuple  $(p^*, T^*)$  guaranteeing that the total cost of the union of all selected teams does not exceed B, after sorting the tuples in descending order of *mergingRatio*.

#### Algorithm 4. PROJECTFIRST

Input: expert-skill-project tripartite graph  $\mathcal{G} = (\mathcal{X}, \mathcal{S}, \mathcal{P}, C, \delta, F)$ , where  $n = |\mathcal{X}|, m = |\mathcal{S}|$  and  $l = |\mathcal{P}|, B$ Output: a project-team map 1 Initialize an empty map projectTeamMap,  $temp \leftarrow \mathcal{P}$ ,  $cost \leftarrow 0$ ;  $2 c(e) \leftarrow \delta(e)$ , for each  $e \in \mathcal{X}$ ; 3 while  $cost \leq B$  and  $temp \neq \emptyset$  do Initialize an empty map tptm; 4 5for each p in temp do  $pce \leftarrow \{e \in \mathcal{X} | c(e) > 0, N(e) \cap N(p) \neq \emptyset\};$ 6  $T \leftarrow \text{GREEDYSETCOVER}(\{N(e) | e \in pce\}, N(p));$ 7 8 if  $T \neq$  None do 9  $tptm \leftarrow tptm \cup \{(p,T)\};$ 10 else  $p \leftarrow \text{None};$ 11 12  $temp \leftarrow \{p \in temp | p \neq \text{None}\};$ if  $projectTeamMap = \emptyset$  do 13 $\frac{F(p^*)}{C(T^*)} \leftarrow \max_{(p,T) \in tptm} \frac{F(p)}{C(T)}$ 14 Add  $(p^*, T^*)$  to project Team Map; 15 $cost \leftarrow C(T^*), temp \leftarrow temp - \{p^*\}, c(e) \leftarrow c(e) - 1, \text{ for any } e \in T^*;$ 16 else 17 18 $curProject \leftarrow \emptyset, curTeam \leftarrow \emptyset;$ for each (p', T') in projectTeamMap do 1920 $curProject \leftarrow curProject \cup \{p'\}, curTeam \leftarrow curTeam \cup T';$ 21for each (p, T) in tptm do  $mergingRatio(p,T) \leftarrow \frac{F(curPorject \cup \{p\})}{C(curTeam \cup \{T\})};$ 2223Sort the tuple (p, T) in *tptm* in descending order of *mergingRatio*(p, T); 24Select the first tuple  $(p^*, T^*)$  from tptm with  $C(curTeam \cup \{T^*\}) \leq B$ ; 25if  $(p^*, T^*) \neq$  None do 26Add  $(p^*, T^*)$  to projectTeamMap; 27 $c(e) \leftarrow c(e) - 1$ , for any  $e \in T^*$ ; 28 $cost \leftarrow C(curTeam \cup \{T^*\}), temp \leftarrow temp - \{p^*\};$ 29else 30 Break 31 Return projectTeamMap;

In PROJECTFIRST, lines  $6 \sim 12$  spend at most O(mnl) on obtaining project-team tuples because the worst-case time complexity of GREEDYSETCOVER is O(mn) according to the conclusion in [2]. Lines  $14 \sim 18$  spend no more than O(l) on generating a seed. Lines  $20 \sim 22$  spend no more than O(l) on computing mergingRatio of each (p,T). Lines 23~24 spend at most  $O(l \log l)$  on selecting the first feasible tuple via the order-first selection strategy. Above all, lines  $4 \sim 30$  take at most  $O((mn + \log l)l)$  to get a participation-constrained profit-maximizing team. Therefore, the worst-case time complexity of PROJECT-FIRST is  $O((mn + \log l)l)$ . For PROJECTFIRST, we only need to store the members in  $\mathcal{X}$ , the skill sets associated with projects in  $\mathcal{P}$  and the experts in  $\mathcal{X}$ . Therefore, the worst-case space complexity of PROJECTFIRST is O(n+ml+nm).

#### 5.2 Expert-Removing Algorithm

This subsection is on an expert-removing approach to form a profit-maximizing feasible team. It takes a universal project-team map as its input. Therefore, we first obtain the universal project-team map through a preprocessing procedure which is introduced in Subsection 5.2.1. Then, we select experts from the universal project-team map according to their influence defined in Definition 7. Note in particular that the participation constraint is considered when constructing the universal project-team map. Therefore, we do not need to consider it in the selection process.

#### 5.2.1 Preprocessing

We first initialize projectTeamMap that associates a project with its corresponding team. For each project  $p \in \mathcal{P}$ , the expert e such that c(e) > 0 and  $N(e) \cap$   $N(p) \neq \emptyset$  is added to the candidate expert set *pce* of *p*. The nonredundant team *T* that covers *p* is computed by GREEDYSETCOVER. If T = None, *p* cannot be finished. Otherwise, let  $ratio(p) = \frac{F(p)}{C(T)}$  for any available project *p*. We select  $(p^*, T^*)$  such that  $\frac{F(p^*)}{C(T^*)} = \max_{p \in \mathcal{P}} \frac{F(p)}{C(T)}$ . The entry  $(p^*, T^*)$  is added to projectTeamMap. At this time,  $p^*$  is accomplishable. The capacities of experts in  $T^*$  decrease by 1. This loop does not end until all accomplishable projects are traversed. The algorithm PREPROCESS (Algorithm 5) describes this preprocessor procedure.

In PREPROCESS, we only consider the participation constraint and ignore the cost constraint. Hence, the map yielded by PREPROCESS is a universal projectteam map. Thereby, the influence of each expert in *projectTeamMap* can be computed according to (3). Below, we introduce an algorithm, which takes a universal project-team map as input, to form a participationconstrained profit-maximizing team.

# 5.2.2 Forming a Balanced Team

PREPROCESS yields a universal project-team map projectTeamMap. According to Definition 1, each project in the map is covered by a unique specified team. Without loss of generality, let  $U = \{(p_1, T_1), \dots, (p_s, T_s)\}$  and  $T = \bigcup_{i=1}^s T_i$ .

Below, we repeat the following procedure until the total cost of the current T exceeds B. First, we compute the influence of each expert  $e \in T$  by (3). Then, the expert  $e^*$  with the minimum influence is selected, and correspondingly the entry  $(p_i, T_i)$  such that  $e^* \in T_i$  is removed from U. T is updated by the union of the teams in the current U. Algorithm ERA (Algorithm 6) describes this procedure.

Lines  $2\sim3$  spend at most O(|U|) on computing the set of experts in U. Lines  $5\sim15$  spend no more than

### Algorithm 5. PREPROCESS

Input: expert-skill-project tripartite graph  $\mathcal{G} = (\mathcal{X}, \mathcal{S}, \mathcal{P}, C, \delta, F)$ , where  $n = |\mathcal{X}|, m = |\mathcal{S}|$  and  $l = |\mathcal{P}|$ Output: a project-team map 1 Initialize an empty map projectTeamMap, temp  $\leftarrow \mathcal{P}, c(e) \leftarrow \delta(e)$  for each  $e \in \mathcal{X}$ ; 2 while  $temp \neq \emptyset$  do 3 for each p in temp do  $pce \leftarrow \{e \in \mathcal{X} | c(e) > 0, N(e) \cap N(p) \neq \emptyset\};$ 4  $T \leftarrow \text{GREEDYSETCOVER}(\{N(e) | e \in pce\}, N(p));$ 5if  $T\neq \mbox{None}$  do 6  $ratio(p) \leftarrow \frac{F(p)}{C(T)}$  $\overline{7}$ 8 else 9  $p \leftarrow \text{None};$  $temp \leftarrow \{p \in temp | p \neq \text{None}\}, \frac{F(p^*)}{C(T^*)} \leftarrow \max_{p \in temp} \frac{F(p)}{C(T)};$ Add  $(p^*, T^*)$  to projectTeamMap,  $c(e) \leftarrow c(e) - 1$  for any  $e \in T^*$ ,  $temp \leftarrow temp - \{p^*\};$ 1011 12 Return projectTeamMap;

# Algorithm 6. ERA

Input: universal project-team map projectTeamMap Output: a project-team map 1 expertSet  $\leftarrow \emptyset$ ; 2 for each p in projectTeamMap do 3  $expertSet \leftarrow expertSet \cup projectTeamMap(p);$  $4 \ cost \leftarrow C(expertSet);$ 5 while cost > B do Initialize an empty dominated project set  $\mathcal{D}$ ; 6 7 for each e in expertSet do for each p in projectTeamMap do 8 if  $e \in projectTeamMap(p)$  do 9  $\mathcal{D}(e) \leftarrow \mathcal{D}(e) \cup \{p\};$ 10 Compute influence(e) according to (3); 11  $e^* \leftarrow \arg\min_{e \in expertSet} influence(e)$ ; Remove each  $p \in \mathcal{D}(e^*)$  and its value from projectTeamMap; 1213  $expertSet \leftarrow \emptyset$ , and update expertSet with the experts in projectTeamMap; 14  $cost \leftarrow C(expertSet);$ 15 Return projectTeamMap

 $O(|T| \times |U|^2)$  on generating a cost-constrained map that associates a project with its corresponding team. Therefore, the worst-case time complexity of ERA is  $O(|T| \times |U|^2)$ . For ERA, we only need to store the projects in U and the expert sets associated with them. Therefore, the worst-case space complexity of ERA is  $O(|U| + \sum_{i=1}^{s} |T_i|)$ .

Below, we exemplify algorithm ERA with example 1. The complete process is shown in Fig.3. The project-team map yielded by PREPROCESS is  $\{(p_1, \{e_1\}), (p_2, \{e_1, e_3\})\}$ . This is a universal project-team map satisfying the participation constraint for the example in Fig.1. Then, we remove the entry (p, T) such that the expert with the minimum influence is in T until the total cost of remaining experts does not exceed B. In this example, the participation-constrained profit-maximizing within the specified budget B is the universal project-team map yielded by PREPROCESS, because  $C(e_1) + C(e_3) \leq B$ .

#### 6 Experimental Evaluation

In this section, we conduct some experiments to evaluate the performance of our algorithms, including effectiveness evaluation, feasibility evaluation and scalability evaluation. Python is used to implement our algorithms on the laptop with Intel<sup>®</sup> Core<sup>TM</sup> i3 2.53 GHz CPU and 2.99 G memory.

# 6.1 Datasets

We conduct these experiments on two real datasets. Their profiles are shown in Table 1. Freelancer is collected from www.freelancer.com and Guru is collected from www.guru.com. For the PARTCONSCH problem, the capacities of experts on these datasets are a random integer ranging from 1 to 3.

Table 1. Two Real-World Datasets

	Freelancer	Guru
$ \mathcal{X} $	1 1 6 0	4650
$ \mathcal{P} $	408	671
$ \mathcal{S} $	464	4183

### 6.2 Load Analysis

IMPROVEDCH aims to find a profit-maximizing team whose cost does not exceed the specified budget B. Obviously, it does not consider the loads of experts. This may lead to the overwork of some experts. Algorithm INFLUENCE is used to solve the problem IM-PROVEDCH. It can specify a corresponding team for each project. Thereby, we can compute the load of each expert. Table 2 and Table 3 illustrate the minimum and the maximum loads of experts in the team yielded by algorithm INFLUENCE on Freelancer and Guru as B increases respectively.

Table 2. Load Analysis on Freelancer as B Increases

B	Min Load	Max Load
50	7	38
65	10	42
80	9	47
95	15	72
110	13	78

Table 3. Load Analysis on Guru as B Increases

В	Min Load	Max Load
50	25	171
100	7	280
150	5	318
200	3	353
250	3	389

#### 148

Yu Zhou et al.: On Participation Constrained Team Formation



Fig.3. Illustration of algorithm ERA.

According to Table 3, the minimum load is equal to the specified load 3 when B = 200, 250, whereas the maximum load is far larger than 3 whatever B equals. This suggests that there are always some experts in the final team assigned to too many projects simultaneously. This means the load unbalance always happens. As shown in Table 2, the minimum load is larger than the specified load 3 whatever B equals. Therefore, all of the experts in Freelancer are assigned to too many projects.

# 6.3 Effectiveness Analysis

INFLUENCE is used to solve problem IMPROVEDCH. PROJECTFIRST and ERA are used to solve PARTCONsCH. In this experiment, we evaluate the effectiveness of these algorithms in terms of the profits and the cost performance made by our proposed algorithms. Note in particular that the cost performance equals  $\frac{F(P^*)}{C(T^*)}$ , where  $T^*$  is the team yielded by our proposed algorithms and  $P^*$  is the profit made by  $T^*$ .

# 6.3.1 For INFLUENCE

According to Lemma 1, IMPROVEDCH is equivalent to CLUSTERHIRE. Therefore, the existing algorithms for CLUSTERHIRE, including EXPERTGREEDY and PROJECTGREEDY, and CLIQUEGREEDY, can be viewed as the baseline. Fig.4 shows the profits and the cost performances made by these algorithms on the two real datasets under different budgets B. Figs.4(a) and 4(b) indicate that the profits and profit-cost ratios made by INFLUENCE are larger than those made by the others on Freelancer. Figs.4(c) and 4(d) indicate that the profits and profit-cost ratios made by INFLUENCE are higher than those made by the others on Guru. In summary, INFLUENCE is better than the three state-ofthe-art algorithms in terms of the profits and the cost performance (the profit-cost ratios).

# 6.3.2 For PROJECTFIRST and ERA

This paper provides two algorithms for PARTCON-SCH, namely PROJECTFIRST and ERA and Fig.5 shows the comparisons of these two algorithms. Figs.5(a) and 5(b) show the profits and profit-cost ratios under different B on Freelancer, and Figs.5(c) and 5(d) show the profits and profit-cost ratios under different B on Guru. They demonstrate that algorithm ERA performs better than algorithm PROJECTFIRST in terms of the profits and the cost performance (profit-cost ratios).

Fig.6 shows the profits and the cost performances of ERA and PROJECTFIRST under different sizes for  $|\mathcal{X}|$ ,  $|\mathcal{S}|$  and  $|\mathcal{P}|$  respectively. The profits are shown in Figs.6(a)~6(c), and the ratios are shown Figs.6(d)~6(f). According to these figures, ERA is better than PROJECTFIRST in terms of the profits and ra-



Fig.4. Comparing INFLUENCE with state-of-the-art algorithms. (a) Profits on Freelancer. (b) Ratios on Freelancer. (c) Profits on Guru. (d) Ratios on Guru.



Fig.5. Comparing ERA with PROJECTFIRST under different B. (a) Profits on Freelancer. (b) Ratios on Freelancer. (c) Profits on Guru. (d) Ratios on Guru.



Fig.6. Comparing ERA with PROJECTFIRST on Guru under different  $|\mathcal{X}|$ ,  $|\mathcal{P}|$  and  $|\mathcal{S}|$ . (a) Profits under different  $|\mathcal{X}|$ . (b) Profits under different  $|\mathcal{P}|$ . (c) Profits under different  $|\mathcal{S}|$ . (d) Ratios under different  $|\mathcal{X}|$ . (e) Ratios under different  $|\mathcal{P}|$ . (f) Ratios under different  $|\mathcal{S}|$ .

tios. In addition, we discover that the profits and ratios of ERA under different  $|\mathcal{X}|$  have a little fluctuation.

## 6.4 Time Efficiency Analysis

In this subsection, we analyze the time efficiency of our proposed algorithms under different B,  $|\mathcal{X}|$ ,  $|\mathcal{S}|$  and  $|\mathcal{P}|$ .

## 6.4.1 For INFLUENCE

Fig.7 shows the comparison of the running time of INFLUENCE and the existing algorithms on Guru. Note that the running time of algorithm UPTMA has been incorporated into that of INFLUENCE. Fig.7 reveals that the running time of INFLUENCE is still lower than that of the others under different B,  $|\mathcal{X}|$ ,  $|\mathcal{S}|$  and  $|\mathcal{P}|$ , even if it includes the running time of UPTMA.

Combining the effectiveness of INFLUENCE in Subsection 6.3.1, we conclude that INFLUENCE performs better than the state-of-the-art algorithms either in terms of the effectiveness or in terms of the time efficiency.

# 6.4.2 For PROJECTFIRST and ERA

Fig.8 shows the comparison of the running time of PROJECTFIRST and ERA. It reveals the running time

of PROJECTFIRST and ERA under different B,  $|\mathcal{X}|$ ,  $|\mathcal{S}|$ and  $|\mathcal{P}|$ . Note that the running time of PREPROCESS has been incorporated into the running time of ERA. Fig.8(a) demonstrates that the running time of PRO-JECTFIRST is larger than that of ERA as B = 200, 250. In Figs.8(b)~8(d), the running time of ERA is always longer than that of PROJECTFIRST. This is because PREPROCESS spends too much time on computing the universal project-team map. Combining the conclusion in Fig.6, we know ERA performs better than PROJECT-FIRST in terms of the profit and the cost performance, whereas it takes more time than the latter. That is to say, they focus on different aspects. One is the time efficiency and the other is the effectiveness.

# 6.5 Memory Usage Analysis

In this subsection, we analyze the memory usage of our proposed algorithms line by line on Freelancer. The python module "memory\_profiler" is used to monitor the memory consumption of the algorithms as well as the line-by-line analysis of memory consumption for each algorithm.

Figs.9(a)~9(d) show the line-by-line memory consumption of INFLUENCE, EXPERTGREEDY, PROJECT-GREEDY and CLIQUEGREEDY respectively. According



Fig.7. Running time of INFLUENCE and state-of-the-art algorithms on Guru under different B,  $|\mathcal{X}|$ ,  $|\mathcal{S}|$  and  $|\mathcal{P}|$ . (a) Different B. (b) Different  $|\mathcal{X}|$ . (c) Different  $|\mathcal{S}|$ . (d) Different  $|\mathcal{P}|$ .



Fig.8. Running time of PROJECTFIRST and ERA on Guru under different B,  $|\mathcal{X}|$ ,  $|\mathcal{S}|$  and  $|\mathcal{P}|$ . (a) Different B. (b) Different  $|\mathcal{X}|$ . (c) Different  $|\mathcal{S}|$ . (d) Different  $|\mathcal{P}|$ .



Fig.9. Line-by-line memory usage of (a) INFLUENCE, (b) EXPERTGREEDY, (c) PROJECTGREEDY, (d) CLIQUEGREEDY, (e) ERA and (f) PROJECTFIRST.

to them, the memories of INFLUENCE and the state-ofthe-art algorithms are not very volatile. They fluctuate from 121 MiB to 125 MiB (Mega Binary Byte). This is because their space complexity only depends on the size of the expert-skill-project tripartite graph. Figs.9(e) and 9(f) show the line-by-line memory consumption of ERA and PROJECTFIRST respectively. According to them, the memories of these two algorithms are not very volatile. They fluctuate from 124.8 MiB to 125 MiB. This is because their space complexity only depends on the size of the expert-skill-project tripartite graph.

# 7 Conclusions

In this paper, we studied the problem of forming a cost-constrained profit-maximizing team without participation constraint, termed IMPROVEDCH, and the problem of forming a participation-constrained profitmaximizing team within the specified budget, termed PARTCONSCH. We proved that IMPROVEDCH was equivalent to problem CLUSTERHIRE, and proposed an efficient and effective algorithm for it, which was based on an expert-removing strategy on a universal projectteam map. To the best of our knowledge, the problem PARTCONSCH was first proposed by us. We proved that its decision version was NP-complete and NP-hard to approximate, and devised two algorithms, named PROJECTFIRST and ERA for it. The former takes the order-first selection strategy. The latter takes a universal project-team map as input, and removes the expert with the lowest influence until the total cost is less than the specified budget.

## References

- Golshan B, Lappas T, Terzi E. Profit-maximizing cluster hires. In Proc. the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 2014, pp.1196-1205.
- [2] Chvatal V. A greedy heuristic for the set-covering problem. Mathematics of Operations Research, 1979, 4(3): 233-235.
- [3] Zzakarian A, Kusiak A. Forming teams: An analytical approach. *IIE Trans.*, 1999, 31(1): 85-97.
- [4] Lappas T, Liu K, Terzi E. Finding a team of experts in social networks. In Proc. the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, June 28-July 1, 2009, pp.467-476.

- [5] Anagnostonpoulos A, Becchetti L, Castillo C, Gionis A, Leonard S. Power in unity: Forming teams in large-scale community systems. In Proc. the 19th ACM International Conference on Information and Knowledge Management, October 2010, pp.599-608.
- [6] Anagnostopoulos A, Becchetti L, Castillo C, Gionis A, Leonardi S. Online team formation in social networks. In Proc. the 21st International Conference on World Wide Web, April 2012, pp.839-848.
- [7] Kargar M, An A. Discovering top-k teams of experts with/without a leader in social networks. In Proc. the 20th ACM International Conference on Information and Knowledge Management, October 2011, pp.985-994.
- [8] Datta S, Majumder A, Naidu K. Capacitated team formation problem on social networks. In Proc. the 18th International Conference on Knowledge Discovery and Data Mining, August 2012, pp.1005-1013.
- [9] Roy S B, Lakshmanan L S V, Liu R. From group recommendations to group formation. In Proc. the ACM SIG-MOD International Conference on Management of Data, May 31-June 4, 2015, pp.1603-1616.
- [10] Li L Y, Tong H H, Cao N, Ehrlich K, Lin Y R, Buchler N. Replacing the irreplaceable: Fast algorithms for team member recommendation. In Proc. the 24th International Conference on World Wide Web, May 2015, pp.636-646.
- [11] Li C T, Shan M K. Composing activity groups in social networks. In Proc. the 21st International Conference on Information and Knowledge Management, October 29-November 2, 2012, pp.2375-2378.
- [12] Li K, Lu W, Bhagat S, Lakshmanan L V S, Yu C. On social event organization. In Proc. the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 2014, pp.1206-1215.
- [13] Feng K Y, Cong G, Bhowmick S S, Ma S. In search of influential event organizers in online social networks. In Proc. the ACM SIGMOD International Conference on Management of Data, June 2014, pp.63-74.
- [14] Shuai H H, Yang D N, Yu P S et al. Willingness optimization for social group activity. Proceedings of the VLDB Endowment, 2013, 7(4): 1035-1045.
- [15] She J Y, Tong Y X, Chen L, Cao C C. Conflict-aware eventparticipant arrangement. In Proc. the 31st IEEE International Conference on Data Engineering, April 2015, pp.735-746.
- [16] She J Y, Tong Y X, Chen L. Utility-aware social eventparticipant planning. In Proc. the ACM SIGMOD International Conference on Management of Data, May 31-June 4, 2015, pp.1629-1643.
- [17] Shen C Y, Yang D N, Lee W C, Chen M S. Maximizing friend-making likelihood for social activity organization. In Proc. the 19th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining, May 2015, pp.3-15.
- [18] Armenatzoglou N, Pham H, Ntranos V, Papadias D, Shahabi C. Real-time multi-criteria social graph partitioning: A game theoretic approach. In Proc. the ACM SIGMOD International Conference on Management of Data, May 31-June 4, 2015, pp.1617-1628.
- [19] Tong Y X, She J Y, Meng R. Bottleneck-aware arrangement over event-based social networks: The max-min approach. World Wide Web — Internet & Web Information Systems, 2016, 19(6): 1151-1177.

J. Comput. Sci. & Technol., Jan. 2017, Vol.32, No.1

- [20] Dumais S T, Nielsen J. Automating the assignment of submitted manuscripts to reviewers. In Proc. the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, June 1992, pp.233-234.
- [21] Karimzadehgan M, Zhai C X, Belford G. Multi-aspect expertise matching for review assignment. In Proc. the 17th ACM Conference on Information and Knowledge Management, October 2008, pp.1113-1122.
- [22] Kou N M, Hou H L, Mamoulis N, Gong Z. Weighted coverage based reviewer assignment. In Proc. the ACM SIG-MOD International Conference on Management of Data, May 31-June 4, 2015, pp.2031-2046.
- [23] Mimno D, McCallum A. Expertise modeling for matching papers with reviewers. In Proc. the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 2007, pp.500-509.



Yu Zhou is a Ph.D. student in the School of Computer Science and Technology at Xidian University, Xi'an. His research interests include data mining, statistical machine learning and heterogeneous information network.



Jian-Bin Huang received his Ph.D. degree in pattern recognition and intelligent systems from Xidian University, Xi'an, in 2007. He is a professor in the School of Software at Xidian University, Xi'an. His research interests include data mining and knowledge discovery.



Xiao-Lin Jia received her Ph.D. degree in the computer software and theory from Xi'an Jiaotong University, Xi'an, in 2006. She is a senior engineer in the Department of Computer Science and Technology at Xi'an Jiaotong University, Xi'an. Her interests include distributed system, software engineering

and wireless sensor network.



He-Li Sun received her Ph.D. degree in computer science from Xi'an Jiaotong University, Xi'an, in 2011. She is an associate professor in the Department of Computer Science and Technology at Xi'an Jiaotong University, Xi'an. Her research interests include data mining and information retrieval.