Batch Crowdsourcing for Complex Tasks Based on Distributed Team Formation in E-Markets

Jiuchuan Jiang, Kai Di, Bo An, *Member, IEEE*, Yichuan Jiang, *Senior Member, IEEE*, Zhan Bu, and Jie Cao

Abstract—Team formation has been extensively studied for complex task crowdsourcing in E-markets, in which a set of workers are hired to form a team to complete a complex task collaboratively. However, existing studies have two typical drawbacks: 1) each team is created for only one task, which may be costly and cannot accommodate crowdsourcing markets with a large number of tasks; and 2) most existing studies form teams in a centralized manner by the requesters, which may place a heavy burden on requesters. In fact, we observe that many complex tasks at real-world crowdsourcing platforms have similar skill requirements and workers are often connected through social networks. Therefore, this paper explores distributed team formation-based batch crowdsourcing for complex tasks to address the drawbacks in existing studies, in which similar tasks can be addressed in a batch to reduce computational costs and workers can self-organize through their social networks to form teams. To solve such an NP-hard problem, this paper presents two approaches: one is to form a fixed team for all tasks in the batch; the other is to form a basic team that can be dynamically adjusted for each task in the batch. In comparison, the former approach has lower computational complexity but the latter approach performs better in reducing the total payments by requesters. With the experiments on a real-world dataset comparing with previous benchmark approaches, it is shown that the presented approaches have better performance in saving the costs of forming teams, payments by requesters, and communication among team members; moreover, the presented approaches have higher success rate of tasks and much better scalability.

Index Terms—Batch crowdsourcing, complex tasks, team formation, social network, distributed manner

1 Introduction

Crowdsourcing refers to outsource tasks to a group of workers chosen from a population [1], [2]. Although many traditional crowdsourcing platforms, such as Amazon's Mechanical Turk, are used for simple tasks, now there are some crowdsourcing platforms oriented to complex tasks, such as Upwork. Therefore, crowdsourcing of complex tasks has attracted significant attention [4], [5], [6].

A popular approach to crowdsource a complex task is to decompose the complex task into a workflow of sub-tasks and then the decomposed sub-tasks are allocated to individual workers [16], [17]. The task decomposition is implemented usually by the requesters [16] or sometimes based on the crowd [17]. However, this approach requires the significant decomposition capability of the requesters [33] or managing the extensive work of crowd [17]. To this end, team formation has been explored recently, in which workers with different skills are hired by the requester to form a team to complete a complex task collaboratively [4], [5], [6], [7], [8].

In general, existing related studies on team formation-based crowdsourcing typically have the following two characteristics. *First*, they are *individual task-specific*, i.e., a new team must be formed from scratch each time when a complex task is published; thus, this method may be costly and cannot adapt to general crowdsourcing markets with a large number of tasks. *Second*, they are *requester-focused*, i.e., team formation is often implemented in a centralized manner by the requester. Although a few studies [4] presented a self-organized strategy where the hired workers can select the teammates themselves, the requesters must undertake a large amount of work around the hiring and training of the initial candidate workers. Such a requester-focused team formation approach may be extremely burdensome to the requesters; moreover, the requesters sometimes may not have enough expertise to justify whether a bidding worker has qualified professional skills.

Our solution to address the above drawbacks in team formation-based crowdsourcing is motivated by the following observations in popular crowdsourcing websites.

1) *The skill requirements of many tasks in the same category at a crowdsourcing website are often similar* (the details can be seen in Section 3.1). By analyzing the tasks from the typical categories at a popular crowdsourcing website,

- *Jiuchuan Jiang, Zhan Bu, and Jie Cao are with the Jiangsu Provincial Key Laboratory of E-Business, School of Information Engineering, Nanjing University of Finance and Economics, Nanjing 210023, China. E-mail: {jcjiang, zhanbu}@nufe.edu.cn, caojie6909129@163.com.*
- *Kai Di and Yichuan Jiang are with the School of Computer Science and Engineering, Southeast University, Nanjing 211189, China. E-mail: {dikai, yjiang}@seu.edu.cn.*
- *Bo An is with the School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798. E-mail: boan@ntu.edu.sg.*

Manuscript received 13 June 2020; revised 14 Feb. 2022; accepted 7 Mar. 2022. Date of publication 22 Mar. 2022; date of current version 11 July 2022.
The work was supported in part by the National Key Research and Development Program of China under Grant 2019YFB1405000, in part by the National Natural Science Foundation of China under Grants 62076060, 61932007, 92046026, and 71871109, and in part by the Natural Science Foundation of Jiangsu Province of China under Grant BK20201394.
(Corresponding author: Jiuchuan Jiang.)
Recommended for acceptance by A. Iosup.
Digital Object Identifier no. 10.1109/TPDS.2022.3161019

Upwork, we find that the tasks are often similar and have overlapping skill requirements; moreover, by comparing the factual completion time and predefined deadline of tasks at Upwork, we find that most tasks have very abundant time to be completed. Therefore, it would be useful to integrate the tasks with similar skill requirements into a batch and outsource them to a team of workers, which can both reduce the team formation cost and make team members contemporaneously undertake more than one task to reduce real execution cost of tasks since the partial execution results of one task can be reused by another similar task undertaken by the same worker team.

2) *Workers are often connected through social networks and cooperate with each other*. Some notable studies [9], [10], [12] have shown that workers often communicate via phone, forums, Facebook, or in person to share information and cooperate through social networks. Therefore, it is possible that workers can self-organize team formation through the social networks. Moreover, this paper attempts to explore a distributed approach different from the study in [4]: the requester needs not undertake hiring and training of the candidate workers but only needs to assign a worker for initializing the team formation for a batch of tasks.

Then, this paper investigates batch crowdsourcing based on distributed team formation; the optimization objective is to minimize the total cost of team formation-based crowdsourcing for a batch of tasks in a social network, which is proven to be a NP-hard problem.

To solve such an NP-hard problem, at first we present an approach to form a fixed team for all tasks in the batch. This approach has lower computational complexity, but requesters need to pay more because each team member should be paid even such member does not execute all tasks in the batch. Therefore, to reduce the total payments by requesters, we then present another approach to form a basic team that can be dynamically adjusted for each task in the batch. In both approaches, each team member's factual income can be improved although the total payments of requesters are reduced, because each team member undertakes more tasks contemporaneously.

In our approaches, first, a worker is selected by the requester to initialize the team formation process to select another qualified worker to join the team. The existing team members can select other qualified workers to join the team step by step until a suitable team for the batch of tasks is formed. Therefore, the requester only needs to select the first initiator worker, which avoids imposing all team formation computation loads on the requester.

With the theoretical analyses and experiments compared with previous benchmark approaches, it is shown that our approaches have better performance in reducing cost and improving higher success rate of tasks.

The rest of this paper is organized as follows. In Section 2, we compare our work with the related work; in Section 3, we present the motivation and problem description; in Sections 4 and 5, we present the distributed formation approaches for fixed and dynamic teams respectively; in Section 6, we provide experimental results; in Section 7, we conclude our paper.

2 Related Work

2.1 Crowdsourcing of Complex Tasks

Based on the task complexity concept in [31], a complex task is constituted by many sub-tasks and the workflows among the sub-tasks, and then it requires multiple higher-level skills. Traditional crowdsourcing technology is difficult to support the task complexity. One reason is that workers often execute tasks independently and thereby the interdependence among sub-tasks cannot be satisfied; the other reason is that the workers are often not professional and cannot satisfy the higher-level skills of complex tasks.

Recently, there are two types of studies for crowdsourcing complex tasks: decomposition allocation-based crowdsourcing and monolithic allocation-based crowdsourcing.

In the decomposition allocation method, a task will first be decomposed into a flow of simple sub-tasks and then the decomposed sub-tasks will be allocated to individual workers [27], [38], [39], which is mainly used in the following two situations: crowdsourcing systems that are oriented to micro-task markets, such as Amazon's Mechanical Turk, and situations where the workers are non-professional and can only complete simple or micro-tasks. A representative study is conducted by Tran-Thanh *et al.* [16], [17], which proposed an algorithm, BudgetFix, to specify the number of micro-tasks to be allocated at each phase of a workflow and dynamically allocates its budget to each micro-task.

Monolithic allocation-based crowdsourcing means that a complex task is directly allocated as a whole to a professional worker. This method is now used at some complex tasks-oriented websites such as Upwork[1]. The task allocation often considers the following three factors: matching degree between the task's required skills and the worker's skills, the reputation and the reservation wage of the worker. At some crowdsourcing websites such as Crowdworks[2], the requesters may interview with the candidate workers through instant messaging software tools to observe whether the workers can complete the tasks [18].

There are some studies on the batch crowdsourcing of simplex tasks [1], [3], [21], [24]. However, to the best of our knowledge, there are no related works on batch crowdsourcing of *complex* tasks. In related studies, each complex task is outsourced individually and dependently. In comparison, this paper addresses the crowdsourcing of a batch of tasks, i.e., a batch of tasks with similar skill requirements can be allocated to a team of workers.

2.2 Team Formation

Team formation is a general concept in many areas [8], [19]. In many existing studies, the team formation of workers is centrally controlled by the requester. Liu *et al.* [7] presented a method that is implemented through profitable and truthful pricing mechanisms. Kargar *et al.* [15] presented a team formation method to satisfy the two objectives in social networks: finding a team of experts that covers all the required skills for the tasks and minimizing the communication cost between workers in the team. Fathian *et al.* [8] integrated the collaboration network and reliability of people and

1 https://www.upwork.com/, accessed August 1, 2021
2 https://crowdworks.jp/, accessed August 1, 2021

presented an optimization model for the formation of a reliable team.

There are also a small number of studies on self-organized team formation. Lykourentzou *et al.* [4] presented a self-organized strategy where the hired workers can select the teammates themselves. Rokicki *et al.* [22] explored a strategy for team formation in which workers themselves decide on which team they want to participate. However, in existing studies, the requester still must undertake hiring and training of the candidate workers and a team is formed for only one task; in comparison, the requester in our study only needs to assign the worker for initializing the team formation for a batch of tasks.

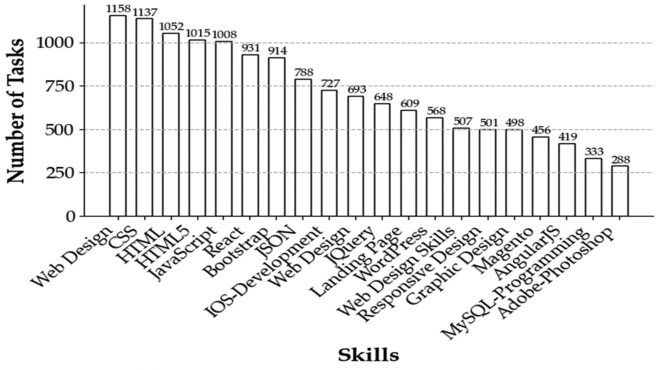
In summary, in existing studies, each team is tailored for a special task but is not efficient in performing other tasks. Although a few studies in engineering management [23] explored the multifunctional team that may be reused for more than one task, they often implemented the team formation in a centralized manner and did not focus on how to address a batch of tasks. In comparison, this paper studies the distributed team formation for a batch of tasks.

2.3 Batching Approach in Distributed Systems and Open Source Software Development

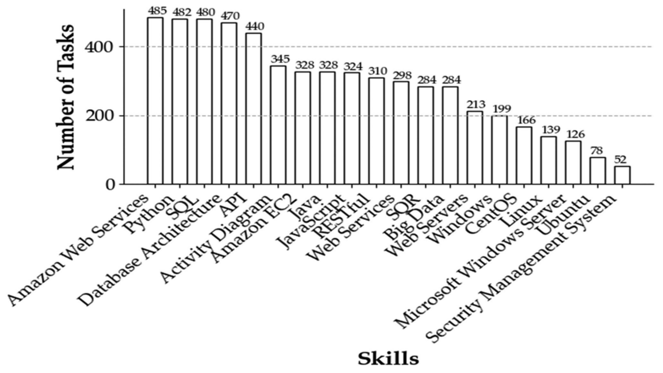
One related series of studies are on the batch job scheduling in distributed systems. Wang *et al.* [29] proposed a batch-processing based coded computing (BPCC) scheme, which can assign proper amount of load to each node to achieve the minimal expected task completion time. Freitas *et al.* [32] proposed a batch task migration approach to improve decentralized global rescheduling, ultimately reducing communication costs and preserving task locality. Alam and Raza [30] proposed a bacterial foraging-based batch scheduling model for distributed systems, which can reduce the node idle time and makespan. In summary, these related studies often aim to design optimal scheduling strategies to minimize makespan or maximize throughput [11]. The optimization objective of this paper is different from those related studies in distributed systems, because this paper aims to reduce the payment by requesters.

Another related series of studies are on the open source software development that is a popular crowdsourcing example [34]. Because developing software from scratch may involve much cost and time, the reuse of software code is one of the central problems [41]. However, it is difficult to design a generic task to reuse in other tasks; moreover, the tasks completed by different workers may often be not compatible. Therefore, this paper presents an alternative method that combines the *similar* tasks into a batch to be performed by the *same* team of workers, which can reduce real execution cost of tasks since the partial execution results of one task can be easily reused by another similar task undertaken by the same workers.

The project team management in open source software development has been investigated. Nan and Kumar [35] found the impact of centralized teams on the project performance for varying software project structural interdependencies. Hahn *et al.* [36] investigated how individuals make decisions about which teams to join and explored how the collaborative network affects developers' choice of new teams to participate in. Those related works only investigated forming a special team for each task; in comparison, this paper explores to form a versatile team for a batch of tasks.



(a) The category of web-mobile-software



(b) The category of IT-networking

Fig. 1. The numbers of tasks requiring some given skills.

3 Motivation and Problem Description

3.1 Motivation

In general, outsourced tasks can be divided into time-sensitive tasks and time-insensitive tasks. Time-sensitive tasks often need to be completed immediately, such as crowdsourced delivery at the Dada[3]. Time-insensitive tasks often have no immediate completion time requirements, such as software development or product design at the Upwork. This paper only focus on the time-insensitive tasks. To illustrate the research problem clearly, we introduce the research motivation by analyzing the data from the typical crowdsourcing website oriented to time-insensitive tasks.

We collected the data of workers and tasks from July 31 –August 6, 2021, from the Upwork that is a very popular website for complex time-insensitive tasks (The collection time window is set as one week because most tasks at the Upwork can be assigned successfully within one week). We wrote a crawler to get the JSON text of the web page. By analyzing the JSON text, the details of the workers and tasks can be gotten.

First, we make statistical analyses on the overlapping skill requirements of tasks in two typical categories at the Upwork: web-mobile-software development and IT-networking. For each category, we select the 20 skills that are

3. https://kuai.imdada.cn/, accessed August 1, 2021.

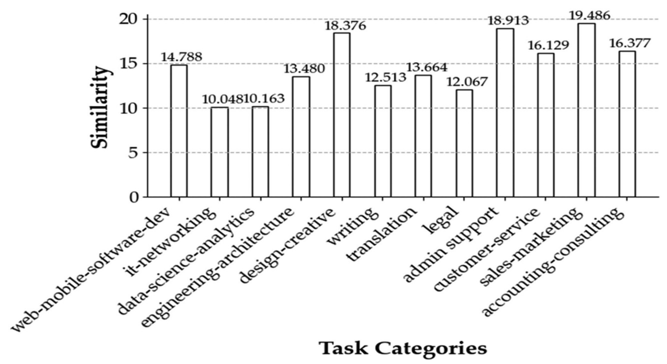


Fig. 2. The similarities of tasks within varying categories at Upwork.

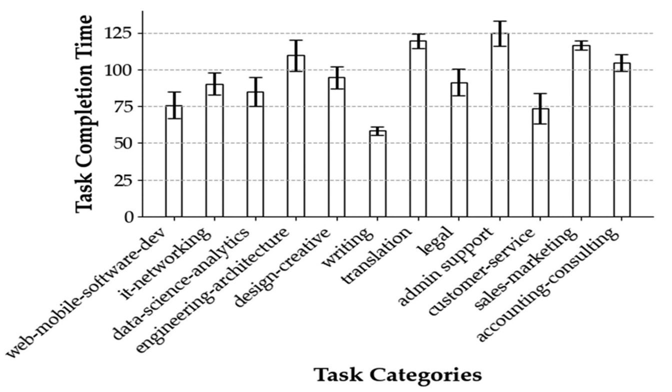


Fig. 3. The factual completion time of tasks at Upwork.

most often required by tasks within such category, which are shown on the x-axis in Fig. 1; the number of tasks requiring each type of skill is represented by the y-axis. We find that each of 20 most popular skills is required by 712 tasks on average in the category of web-mobile-software development and by 291 tasks on average in the category of IT-networking. From these statistical analyses, *it is common that each skill may be required by more than one task*.

Second, we observe the task similarity at the Upwork. Let there be n tasks. We use $S_{t_x}$to denote the set of skills required by task t_x. Then, the similarities among the n tasks can be measured by the following:

$$Similarity = \left(\sum_{x=1}^{n} |S_{t_x}|\right) / \left(|\bigcup\nolimits_{x=1}^{n} S_{t_x}|\right) \tag{1}$$

The higher *Similarity* is, the more similar the n tasks are. We calculate the similarity for each category of the collected 60886 tasks, as shown in Fig. 2. The average similarity of the tasks is 14.67, which denotes that many tasks in the same category is similar. Therefore, *it is possible to pack the similar tasks with highly overlapping skill requirements into a batch and assign them to the same workers possessing these required skills*.

Third, we count the factual completion time of tasks at the Upwork. There are 7726175 registered workers and among them there are 51866 active workers (the Upwork website defines the workers earned more than \$10k as active ones). The active workers completed 91427 tasks totally from July 1 to August 1, 2021. The statistical results of completion time for varying categories of tasks undertaken by active workers are shown in Fig. 3, from which we can see that average completion time is 92.94 hours. On the other hand, after counting the deadlines of tasks at the Upwork, we find that the tasks with fixed price have no strict deadlines and the tasks with hourly wage all have more than 6 months deadlines. Therefore, *it is common that the tasks have very abundant time to be completed; and the tasks can tolerate the waiting time to be completed when they are addressed in a batch*.

Forth, in our previous work [40], it is observed that the time during which workers undertake no more than one task occupy 61.58% of the workers' total time at Upwork[4]. Therefore, most workers undertake only a small number of tasks contemporaneously. Based on this observation, *it is possible that many workers may have leftover time and efforts to undertake more tasks*.

Finally, some notable studies have shown that workers are often connected through social networks. Gray *et al.* [10] observed different crowdsourcing platforms and found that the crowd of workers is a rich network of collaboration in which they often communicate via phone, forums, chat, Facebook, or in person to share information. Yin and Gray [9] specifically observed the collaboration network of workers on the MTurk platform and found that there is a substantial communication network within the crowd of workers, which is related to the workers' usage on online forums. *Based on these observations, it is common that workers often collaborate through social networks. Therefore, workers can self-organize teams through social networks*.

3.2 Problem Description

3.2.1 Discounting Mechanism for a Batch of Tasks

Let there be a batch of tasks, $B = \{t_1, t_2, \ldots, t_{|B|}\}$, where $|B|$ denotes the number of tasks. Let there be two tasks t_x and t_y, $t_x, t_y \in B$. The sets of necessary skills required by t_x and t_y are $S_{t_x} = \{s_{x1}, s_{x2}, \ldots, s_{xm}\}$ and $S_{t_y} = \{s_{y1}, s_{y2}, \ldots, s_{yn}\}$, respectively, where m and n denote the numbers of skills required by t_x and t_y. The skill distance between tasks t_x and t_y is:

$$\delta_{x,y} = 1 - \left(|S_{t_x} \cap S_{t_y}| / |S_{t_x} \cup S_{t_y}|\right) \tag{2}$$

Then, *the diversity of all tasks in B* is defined as:

$$\theta(B) = \left(\sum_{x=1}^{|B|} \sum_{y=1}^{|B|} \delta_{x,y}\right) / (2 \cdot |B|) \tag{3}$$

Discounting is a general market mechanism where payment can be discounted in exchange for cheaper or less satisfactory service [25]. If a worker performs more than one similar task, the real execution cost of each task can be reduced since the partial execution results of one task can be reused by another similar task in the batch; moreover, the requester should also be compensated for waiting for this worker's services when the worker undertakes more than one task contemporaneously. Therefore, discounting can be introduced to reduce the actual payment from requesters to workers, which can make requesters to allocate more tasks to the same workers and workers undertake more tasks contemporaneously.

4. We do not collect the new data in this paper because now the number of tasks that are undertaking by workers cannot be gotten but only the number of tasks completed by workers can be gotten from the Upwork.

A worker will earn more income, and the requester will wait longer if more tasks are allocated to the worker contemporaneously; moreover, the worker will save more in computational costs if the tasks are more similar. Therefore, the larger the batch is or the more similar the tasks in the batch are, the more discounting that can be offered.

Let a batch of tasks, B, be allocated to a worker *at once* and is queueing for execution by that worker, w_i. The real payment to w_i for performing B can be discounted according to the size and diversity of B. Given an original utility $op_{w_i}(t_x)$that the requester of task t_x should pay to w_i, we can define the ***wholesale discounting function*** as follows:

$$P_{w_i}(B) = \sum_{t_x \in B} \left(\psi\left(\frac{|B|}{\theta(B)+1} \right) \cdot op_{w_i}(t_x) \right) \tag{4}$$

where $\boldsymbol{\psi}$ is the discounting function, $0 \leq \boldsymbol{\psi} \leq 1$, and the value of $\boldsymbol{\psi}(X)$ decreases monotonically from 1 to 0 with the increase of X. We use '$\boldsymbol{\theta}(B)+1$' to avoid a situation where the denominator is 0 due to $\boldsymbol{\theta}(B)$ being 0.

If tasks in B are allocated to w_i *one by one*, discounting can be done progressively for each task in B. We can define the ***progressive discounting function*** as follows:

$$P_{w_i}(B) = \sum_{x=1,\cdots,|B|} \left(\psi\left(\frac{x}{\delta_{x-1,x}+1} \right) \cdot op_{w_i}(t_x) \right) \tag{5}$$

where '$\boldsymbol{\delta}_{x\text{-}1,x}+1$' is used to avoid a situation where the denominator is 0 due to $\boldsymbol{\delta}_{x\text{-}1,x}$ being 0; $\boldsymbol{\delta}_{01} = 1$.

3.2.2 Cost of Team Formation-Based Batch Crowdsourcing

In general, the cost of team formation-based crowdsourcing in existing benchmark studies [7], [13], [14], [15] includes three parts: costs of forming teams, payments of requesters, and communication among team members.

1) *The cost of forming teams*. The cost of forming a team can be abstracted as a monotonically increasing function on the number of recruitment and elimination events of team members [13]. We use $f(X)$ to denote a monotonically increasing function whose argument is X.

   Let there be a batch of tasks B. If we form a fixed team for all tasks in B, $W_B = \{w_1,\ldots, w_n\}$, the cost of forming a team for B is $C_{Form}(B) = f(|W_B|)$.

   If we form a dynamic team for B, the members will be dynamically adjusted for each task in the batch. Let there be two tasks: t_x, $t_y \in B$. The teams actually performing t_x and t_y are W_{t_x} and W_{t_y}, respectively; then, the number of elimination and recruitment events of team members from W_{t_x} to W_{t_y} is $|W_{t_x} - W_{t_y}| + |W_{t_y} - W_{t_x}|$. If the execution sequence of tasks in B is from t_1 to $t_{|B|}$, then the cost of team formation for B is:

$$C_{Form}(B) = f\left(|W_{t_1}| + \sum_{x=1,\ldots,|B|-1} \left(|W_{t_x} - W_{t_{x+1}}| + |W_{t_{x+1}} - W_{t_x}| \right) \right) \tag{6}$$

2) *The cost of requesters paying team members*. Let $T_i(B)$ be the set of tasks in B that are actually performed by worker w_i, $T_i(B) \subseteq B$. $\forall t_x \in B$, the budget of t_x *is* b_{t_x}. The set of skills required by t_x is S_{t_x}; let $\overline{S_{t_x}}$ be the set of skills for t_x that are currently lacking; the set of skills of w_i that actually contribute to performing t_x is $\overline{S_{t_x}} \cap S_{w_i}$, where S_{w_i}denotes the set of skills possessed by w_i. Given a budget b_{t_x}provided by the requester of task t_x, b_{t_x}will be distributed to the assigned workers according to their real skill contribution; therefore, the original utility that the requester of task t_x should pay to w_i is determined by $b_{t_x}$and $|\overline{S_{t_x}} \cap S_{w_i}| / |\overline{S_{t_x}}|$.

   If we form a fixed team for all tasks in B, W_B, $\forall w_i \in W_B$, the requester of t_x will pay the following real utilities to w_i by considering w_i's real contribution to t_x and the discounting function:

$$\forall w_i \in W_B : P_{w_i}(t_x) = \begin{cases} \psi\left(\frac{|T_i(B)|}{\theta(T_i(B))+1} \right) \cdot b_{t_x} \cdot \frac{|\overline{S_{t_x}} \cap S_{w_i}|}{|\overline{S_{t_x}}|} if\ t_x \in T_i(B) \\ 0, \qquad else \end{cases} \tag{7}$$

Thus, the total payment of requesters for performing the tasks in B by forming a fixed team is:

$$C_{Pay}(B) = \sum_{t_x \in B} \sum_{w_i \in W_B} P_{w_i}(t_x) \tag{8}$$

If we form a dynamic team for B, i.e., $\forall t_x \in B$, we form W_{t_x} for task t_x. $\forall w_i \in W_{t_x}$, the requester of t_x will pay the following real utilities to w_i:

$$\forall w_i \in W_{t_x} : P_{w_i}(t_x) = \psi\left(\frac{|T_i(B)|}{\theta(T_i(B))+1} \right) \cdot b_{t_x} \cdot \frac{|\overline{S_{t_x}} \cap S_{w_i}|}{|S_{t_x}|} \tag{9}$$

Thus, the total cost of payments of requesters for performing the tasks in B by forming a dynamic team is:

$$C_{Pay}(B) = \sum_{t_x \in B} \sum_{w_i \in W_{t_x}} P_{w_i}(t_x) \tag{10}$$

3) *The communication cost among team members*, $C_{Com}(B)$, which is mainly decided by the size of the team and the distance between members in the social networks.

   Therefore, the total cost for a batch of tasks B is:

$$C(B) = \lambda_1 \cdot C_{Form}(B) + \lambda_2 \cdot C_{Pay}(B) + \lambda_3 \cdot C_{Com}(B)) \tag{11}$$

where λ_1, λ_2, and λ_3 are three weights to measure the relative importance of the three types of costs.

3.2.3 Optimization Objective and Heuristics

Generally, existing studies ensuring the quality of crowdsourcing results often include the following methods: effective task design, redundant allocation of tasks and aggregation of the results, and reputation model [38]. Among them, effective task design is implemented by the requester, which is not the focus of this paper; the redundant allocation is mainly used for simple tasks, because the redundant allocation of complex tasks may produce very high costs. Therefore, we introduce the reputation mechanism; the reputation of a worker is mainly determined by the worker's past experiences in completing tasks [33].

Because the completion of a complex task is determined by many factors, it is difficult to measure the comprehensive performance of the assigned worker; moreover, the quality of a complex task is difficult to measure in reality, thus now many studies only justify whether the complex task is completed successfully. Therefore, the reputation in this paper is related to the history of a worker's successful completing tasks as in previous benchmark work [20]: if a worker completed tasks successfully, his/her reputation increased, and vice versa.

When a worker responds to the request for joining a team, he/she will estimate the completion time for each task in the batch according to the current real situation and his/her experiences; moreover, the estimated completion time of worker for a task can also be estimated by the system. Therefore, the estimated completion time of each task can be achieved before task execution [37].

In this paper, we abstract the team formation-based batch crowdsourcing to minimize the total cost as well as maximize the reputations of team members under four constraints: the skill requirements of all tasks can be satisfied, the payment is more than the reservation wages of team members, all workers can only communicate with their neighbors in the social network, and the estimated completion time of each task cannot exceed the deadline of that task. Let W denote the crowd of workers, R_{w_i} denote the reputation of worker w_i, and γ_{w_i} denote the reservation wage of worker w_i. The optimization objective while forming a fixed team for all tasks in B is as follows:

$$W_{B*} = \underset{W_B}{\arg\min} \left(C(B) \Big/ \sum_{w_i \in W_B} R_{w_i} \right) \tag{12}$$

$$subject\ to:$$

$$(\forall t_x \in B) \wedge \left(\bigcup_{w_i \in W_B} S_{w_i} \supseteq S_{t_x} \right) \tag{13}$$

$$(\forall w_i \in W_B) \wedge (\forall t_x \in B) \wedge (t_x \in T_i(B)) \wedge (P_{w_i}(t_x) \geq \gamma_{w_i}) \tag{14}$$

$$\forall w_i \in W : w_i \text{ can only communicate with his neighbors in the social network} \tag{15}$$

$$(\forall t_x \in B) \wedge time_{W_B}(t_x) \leq d_{t_x} \tag{16}$$

where $time_{W_B}(t_x)$denotes the completion time of task t by W_B, and d_{t_x}denotes the deadline of t_x.

Let W_{t_x} denote the team of workers performing task t_x. Let Π be a set of teams for all tasks in B, $\Pi = \{W_{t_x} \mid \forall t_x \in B\}$. For a worker in any team, $w_{i,}$ the set of tasks assigned to $w_{i,}$ is assumed to B_i. The optimization objective while we form a dynamic team for each task in B is to form the set of teams for B as follows:

$$\prod_{*} = \underset{\prod}{\arg\min} \left(C(B) \Big/ \sum_{t_x \in B} \left(\sum_{(w_i \in W_{t_x}) \wedge (W_{t_x} \in \prod)} R_{w_i} \right) \right) \tag{17}$$

$$Subject\ to:$$

$$(\forall t_x \in B) \wedge (W_{t_x} \in \prod) \wedge \left(\bigcup_{w_i \in W_{t_x}} S_{w_i} \supseteq S_{t_x} \right) \tag{18}$$

$$(\forall t_x \in B) \wedge (W_{t_x} \in \prod) \wedge (\forall w_i \in W_{t_x}) \wedge (P_{w_i}(t_x) \geq \gamma_{w_i}) \tag{19}$$

$$\forall w_i \in W : w_i \text{ can only communicate with his neighbors in the social network} \tag{20}$$

$$(\forall t_x \in B) \wedge time_{W_{t_x}}(t_x) \leq d_{t_x} \tag{21}$$

where $time_{W_{t_x}}(t_x)$denotes the completion time of task t by W_{t_x}, and d_{t_x}denotes the deadline of t_x.

Theorem 1. The team formation problem for a batch of tasks with the optimization objective in Equations (*12*), (*13*), (*14*), (*15*), (*16*) or (*17*), (*18*), (*19*), (*20*), (*21*) is NP-hard.

Proof sketch. Our problem includes three independent sub-problems that can be described to find a team of workers within a social network to satisfy the skill requirements of tasks and minimize the following three factors: the cost of forming teams, the cost of requesters paying team members, and the communication cost among team members. The problem of finding a team of workers in a social network to satisfy the skill requirements of a task and minimize the communication cost has already been proven in previous benchmark studies to be NP-hard [14], [15], [28]. Since our problem involves this well-known NP-hard problem in combination with two other independent problems and satisfying the skill requirements of a batch of tasks, we have Theorem 1. □

To reduce the cost of forming teams, it is preferable to select the workers who cover more skills of the task batch. To reduce the payments by requesters, it is preferable to select workers whose reservation wages are low and workers who have already undertaken more tasks in the batch. To reduce the cost of communication between team members, it is preferable to select workers who are closer in the social network to form a team; therefore, existing team members will find other new teammates from the near to the distant in the social network. Moreover, it is preferable to select workers with higher reputations to ensure the quality of crowdsourcing results.

For the above considerations, this paper presents the concept of the *crowdsourcing value* of a worker to measure the probability of that worker to be hired into a team, which is determined by the following factors: 1) the coverage degree of the worker's skills for the skill requirements of the tasks; 2) the communication distance of the worker with other workers in the social network; 3) the reservation wage of the worker; and 4) the reputation of the worker.

Based on the crowdsourcing value, two heuristic approaches are presented. The first is a distributed formation of a fixed team for all tasks in the batch; the second one is a distributed formation of a dynamic team.

3.2.4 Self-Organization Process of Team Formation Through Social Networks

First, a worker with the maximum probability for approaching the optimization objective is assigned as the initiator. Then, the initiator will send requests to all his/her neighbors through the social network. If a neighbor can respond to the request within a predefined time, the initiator will observe the situation of this neighbor; then, the initiator recruits a worker from all his/her neighbors to join the team for approaching the optimization objective.

Then, the existing team members will send requests to all their neighbors through the social network, and they will recruit one worker from the observed neighbors to join the team for approaching the optimization objective. The recruiting process will be repeated until the skills of all tasks in the batch can be satisfied or all workers in the social network are observed.

4 Distributed Formation of a Fixed Team for Batch Crowdsourcing

4.1 Crowdsourcing Value for a Batch of Tasks

To approach the optimization objective, we define two types of crowdsourcing values: *global crowdsourcing value*, which measures the probability of a worker being assigned as the initiator to form the team; and *local crowdsourcing value*, which measures the probability of a worker being recruited by the existing team members to join the team.

4.1.1 Global Crowdsourcing Value for a Batch of Tasks

The global crowdsourcing value of a worker is generally determined by the following factors:

1) The coverage degree of the worker's skills for the necessary skills required by the batch. Let there be a batch of tasks, B. The set of skills required by all tasks in batch B is $S_B = \cup_{\forall t_x \in B} S_{t_x}$, where S_{t_x} denotes the set of skills required by task t_x. $\forall s_a \in S_B$, we use the number of tasks in the batch that require s_a to denote the weight of s_a for the batch, n_a. Now, we use a binary value to denote whether worker w_i possesses skill s_a: $b_{ia} = 1$ if w_i possesses s_a; and $b_{ia} = 0$ if w_i does not possess s_a. Then, the skill coverage degree of w_i for batch B will consider each skill's weight for the batch, which is:

$$CS_{w_i}(B) = \left(\sum_{\forall s_a \in S_B} (b_{ia} \cdot n_a)\right) \Big/ \left(\sum_{\forall s_a \in S_B} n_a\right) \tag{22}$$

2) *The locality of the worker regarding other workers' skill coverage degrees for the batch*. We will attempt to select more central workers with respect to other workers' skills, i.e., the closer a worker is to other workers that can cover more skills for the batch, the more advantageous the worker's locality will be for being assigned to the batch. Let $CS_{w_i}(B)$denote the coverage degree of worker w_i's skills for batch B. Let d_{ij} denote the distance between workers w_i and w_j in the social network. It is assumed that the set of all workers is W; the locality of w_i for B is defined as follows:

$$Loc_{w_i}(B) = \sum_{j=1\cdots|W|, j\neq i} (d_{ij}/CS_{w_j}(B)) \tag{23}$$

3) *The occupancy rate of the worker's reservation wage on the task's real payment*. Let the reservation wage of w_i be γ_{w_i}. Given an original budget b_{t_x}provided by the requester of task t_x, the occupancy rate of w_i's reservation wage on batch B's payments is:

$$Occ_{w_i}(B) = \left(\sum_{\forall t_x in B} \left(\gamma_{w_i} / \left(\psi\left(\frac{|B|}{\theta(B)+1}\right) \cdot b_{t_x}\right)\right)\right) / |B| \tag{24}$$

If the assigned worker w_i has a lower $Occ_{w_i}(B)$, w_i may distribute more utilities to other team members for executing tasks in B; therefore, it is more probable that w_i will form a successful team for B.

4) *The reputation of worker w_i, R_{w_i}*.

Finally, we have the following definition:

Definition 1. *Global crowdsourcing value of a worker for a batch of tasks.* The global crowdsourcing value of a worker, w_i, for a batch of tasks B is:

$$GV_{w_i}(B) = \frac{\alpha_1 \cdot CS_{w_i}(B) + \alpha_4 \cdot R_{w_i}}{\alpha_2 \cdot Loc_{w_i}(B) + \alpha_3 \cdot Occ_{w_i}(B)} \tag{25}$$

where $\boldsymbol{\alpha}_1 + \boldsymbol{\alpha}_2 + \boldsymbol{\alpha}_3 + \boldsymbol{\alpha}_4 = 1$.

Lemma 1. Let there be two workers, w_i and w_j. If $GV_{w_i}(B) > GV_{w_j}(B)$, w_i can comprehensively approach the optimization objective better than w_j can.

Proof. The optimization objective is to minimize the three types of costs and the reverse of reputations. Let $W_B(w_i)$ be the team formed by w_i for B, and let $W_B(w_j)$ be the team formed by w_j for B. Now $GV_{w_i}(B) > GV_{w_j}(B)$; we can say that the four factors of w_i in (25) are higher than those of w_j if the three factors are endowed with equal weight. $GS_{w_i}(B) > GS_{w_j}(B)$ denotes that w_i's skills can cover the skills of B more than w_j's, and thus $|W_B(w_i)| < |W_B(w_j)|$; therefore, the three types of costs of w_i are less than that of w_j since the cost is influenced by the number of team members. $LOC_{w_i}(B) > LOC_{w_j}(B)$ denotes that w_i can recruit other team members with less numbers and communication distance than w_j; thus, the three types of cost of w_i are also less than that of w_j. $Occ_{w_i}(B) > Occ_{w_j}(B)$ denotes that the cost of paying w_i is less than to w_j. Thus, the third type of cost of w_i is less than that of w_j. All three types of cost of w_i are less than that of w_j and the reputation of w_i is higher than that of w_j; thus, we have Lemma 1. □

4.1.2 Local Crowdsourcing Value for a Batch of Tasks

The local crowdsourcing value of a worker is generally determined by the following four factors:

1) *The coverage degree of the worker's skills for the currently missing skills of the batch*. Let there be a batch, B. Now, let there be an existing team $W_B \subseteq W$. $\forall w_i \in W_B$, the set of skills possessed by w_i is S_{w_i}. Then, the skill coverage degree of w_i with respect to the existing members in W_B for batch B can be defined as follows:

$$CS^{lacking}_{w_i}(B) = \left(\sum_{\forall s_a \in (S_B - \cup_{\forall w_j \in W_B} S_{w_j})} (b_{ia} \cdot n_a)\right) \Big/ \left(\sum_{\forall s_a \in (S_B - \cup_{\forall w_j \in W_B} S_{w_j})} n_a\right) \tag{26}$$

where the symbols S_B, n_a, b_{ia} are the same as those in Eq. (22).

2) *The distance between the worker and the existing team members*. Let $CS_{w_j}(B)$ denote the coverage degree of worker w_j's skills for batch B. The distance between the worker and W_B will consider the existing team members' skill coverage degrees, shown as:

$$D_{w_i}^{team}(B) = \frac{1}{|W_B|}\sum_{\forall w_j \in W_B}(d_{ij}/CS_{w_j}(B)) \tag{27}$$

3) *The reservation wage of the worker*, γ_{w_i}, denotes the minimum wage requirement of worker w_i for one task.
4) *The reputation of worker w_i*, R_{w_i}.

Definition 2. *Local crowdsourcing value of a worker for a batch of tasks.* Let there be an existing non-final team $\mathrm{W_B}$ for batch B, $\mathrm{W_B} \subseteq \mathrm{W}$. The local crowdsourcing value of $\mathrm{w_i}$ perceived by $\mathrm{W_B}$ for B is defined as:

$$LV_{w_i}(B) = \frac{\alpha_1 \cdot CS_{w_i}^{lacking}(B) + \alpha_4 \cdot R_{w_i}}{\alpha_2 \cdot D_{w_i}^{team}(B) + \alpha_3 \cdot \gamma_{w_i}} \tag{28}$$

where $\alpha_1+\alpha_2+\alpha_3+\alpha_4 = 1$.

Lemma 2. Let there be two workers, $\mathrm{w_i}$ and $\mathrm{w_j}$. It is assumed that the set of existing team members formed for task batch B is $\mathrm{W_B}$. If $LV_{w_i}(B) > LV_{w_j}(B)$, $\mathrm{W_B} \cup \{\mathrm{w_i}\}$ can approach the optimization objective better than $\mathrm{W_B} \cup \{\mathrm{w_j}\}$ *can*.

Proof. The proof is similar to that for Lemma 1; here, we skip it to save space. □

4.2 Distributed Formation Algorithm

A worker with the maximum global crowdsourcing value whose reservation wage can be satisfied is assigned as the initiator forming the team. Then, the self-organization process of team formation described in Section 3.2.4 is used through observing the local crowdsourcing values of other workers in the social network. We can set a maximum response time (δ) that denotes the worker should respond within such time, or else a new worker will be selected. Moreover, to satisfy the deadline requirement of a task, the team's estimated completion time of the task cannot exceed its deadline.

However, if all neighbors of existing team members in one round cannot satisfy the requirements, the formation process may end before all skills of the batch can be satisfied, but some un-observed workers still exist in the social network. Therefore, we can revise the formation process by assuming that all observed workers (not only the team members) can help find their neighbors; this method is feasible because workers are often cooperative within the same social network [26].

Let B denote a batch of tasks and W denote the set of workers in the social network. We use S_B to denote the set of skills required by all tasks in B and W_B to denote the team formed for B. The distributed formation of a fixed team for batch crowdsourcing is shown as Algorithm 1. *Res*(w) denotes the response time of worker w. Algorithm 1 is O(| S_B | · | W |).

Theorem 2. Let the team formed for a batch of tasks B using Algorithm 1 be $\mathrm{W_B}$; the initiator worker of $\mathrm{W_B}$ is $\mathrm{w_i}$. It is then assumed that there is another team, $\mathrm{W_B}'$, which is formed by any of the following two methods: another random initiator worker $\mathrm{w_j}$ and the workers in $\mathrm{W_B}$-{$\mathrm{w_i}$} or $\mathrm{w_i}$ and other workers. It is assumed that the skills possessed by $\mathrm{W_B}'$ can fully cover the skill and deadline requirements of B and the reservation wage of each worker in $\mathrm{W_B}'$ can be satisfied. We use $\mathrm{C_{form}(B)\_W_B}$, Cpay (B)_$\mathrm{W_B}$, *and* Ccom(B)_$\mathrm{W_B}$ to denote the three related types of costs in our optimization objective when we form team $\mathrm{W_B}$ to perform B and use R_$\mathrm{W_B}$ to denote the reputations of workers in $\mathrm{W_B}$. Then, we have:

$$\left(\forall W_B' \wedge W_B' \subseteq W \wedge \bigcup_{\forall w_i \in W_B'} S_{w_i} \supseteq S_B \wedge (\forall w_i \in W_B' \wedge \gamma_{w_i} \leq \frac{P_{w_i}(B)}{|B|})\right)$$
$$\Rightarrow (\alpha_1 \cdot C_{Form}(B)\_W_B + \alpha_2 \cdot C_{Pay}(B)\_W_B + \alpha_3 \cdot C_{Com}(B)\_W_B + \alpha_4/R\_W_B)$$
$$\leq (\alpha_1 \cdot C_{Form}(B)\_W_B' + \alpha_2 \cdot C_{Pay}(B)\_W_B' + \alpha_3 \cdot C_{Com}(B)\_W_B' + \alpha_4/R\_W_B')$$

Proof Sketch. If $W_B{}'$ is formed by another random initiator worker w_j and the workers in W_B-{w_i}. According to Algorithm 1, we have $GV_{w_i}(B) > GV_{w_j}(B)$; then, Lemma 1 ensures that w_i can comprehensively approach the optimization objective better than w_j. Because W_B-{w_i} = W_B-{w_j}, we obtain the theorem. If $W_B{}'$ is formed by w_i and other workers in the social network, let the order of w_k in W_B be the same as that of $w_k{}'$ in $W_B{}'$; we then have $LV_{w_k}(B) > LV_{w_k'}(B)$ according to Algorithm 1. Now, Lemma 2 ensures that {w_i}∪{w_k} can approach the optimization objective better than {w_i}∪{$w_k{}'$}. Thus, finally, W_B can approach the optimization objective better than $W_B{}'$. Therefore, we have the theorem. □

Theorem 2 ensures that the optimization objective in Eq. (12) can be approached by Algorithm 1.

5. Distributed Formation of a Dynamic Team for Batch Crowdsourcing

First, we will find the basic set of skills of the batch, and we then form a basic team to satisfy the basic set of skills of the batch. Afterwards, the basic team will be adjusted for each task in the batch.

5.1 Crowdsourcing Value for One Task

5.1.1 *Global Crowdsourcing Value for One Task*

The global crowdsourcing value of a worker for one task is determined by the following factors: 1) the coverage degree of the worker's skills for the necessary skills required by the task; 2) the locality of the worker with respect to other workers' skill coverage degrees for the task; 3) the occupancy rate of the worker's wage on the task's budget; and 4) the reputation of the worker.

Definition 3. *Global crowdsourcing value of a worker for one task.* The global crowdsourcing value of a worker, $\mathrm{w_i}$, for task $\mathrm{t_x}$ is:

$$GV_{w_i}(t_x) = \frac{\alpha_1 \cdot (|S_{w_i} \cap S_{t_x}|/|S_{t_x}|) + \alpha_4 \cdot R_{w_i}}{\alpha_2 \cdot Loc_{w_i}(t_x) + \alpha_3 \cdot (\gamma_{w_i}/b_{t_x})} \tag{29}$$

where $\alpha_1+\alpha_2+\alpha_3+\alpha_4 = 1$, and $Loc_{w_i}(t_x)$ is calculated as follows:

$$Loc_{w_i}(t_x) = \sum_{j=1}^{|W|}(d_{ij}/(|S_{w_j} \cap (S_{t_x} - S_{w_j})|/|S_{t_x}|)) \tag{30}$$

where d_{ij} denotes the distance between workers w_i and w_j in the social network, and W denotes the crowd of workers in the social network.

5.1.2 *Local Crowdsourcing Value for One Task*

The local crowdsourcing value of a worker denotes the probability of the worker to be recruited by the existing team members for a task, which is determined by: 1) the coverage degree of the worker's skills for the current lacking skills by the task; 2) the distance between the worker and the existing team members regarding the team members' skills; 3) the occupancy rate of the worker's wage on the task's budget; and 4) the reputation of the worker.

Algorithm 1. Distributed formation of a fixed team for batch crowdsourcing.

```
1)  Lacking_S_B = S_B; b_1 = 0; W_B = {}; W_temp1 = W; W_temp2 = {};
2)  **While** (b_1 = = 0) **do**: // Assign the initiator worker
3)      $w_* = \arg\max_{\forall w_i \in W_{temp1}}(GV_{w_i}(B))$ ;
4)      **If** Res(w_*)≤$\delta$ **and** $((\forall t_x \in B) \wedge time_{w*}(t_x) \le d_{t_x})$ **and** $(P_{w*}(B)/|B|) \ge \gamma_{w*}$ **and** $S_{w*} \cap$ Lack ing_S_B ≠{}:
5)      {Lacking_S_B = Lacking_S_B-$S_{w*}$; b_1 = 1;}
6)      W_temp1 = W_temp1-{w_*};
7)      **If** (W_temp1 = = {}): b_1 = 1;
6)  b_1 = 0; W_B = {w_*};
7)  W_temp2 = W_temp2 ∪{w_*};//Observed workers
8)  W_candidate = {};//Workers to be observed
9)      **If** (Lacking_S_B = = {}): b_1 = 1;
10) **While** (b_1 = = 0) **do**: //Distributed formation of team
11)     ∀w_i∈W_temp2:
12)         ∀w_j∈Neigh(w_i): //Neighbors of w_i
13)             **If** (w_j ∉ W_temp2): W_candidate = W_candidate∪{w_j};
14)     W_temp2 = W_temp2∪ W_candidate;
15)     b_2 = 0; b_3 = 0
16)     **While** (b_2 = = 0) **do**://Search a qualified neighbor
17)         $w_* = \arg\max_{\forall w_i \in W_{candidate}}(LV_{w_i}(B))$;
18)         **If** Res(w_*)≤$\delta$ **and** $((\forall t_x \in B) \wedge time_{w*}(t_x) \le d_{t_x})$ **and** $(P_{w*}(B)/|B|) \ge \gamma_{w*}$ **and** $S_{w*}$ ∩Lacking_S_B ≠{}:
19)             {b_2 = 1; b_3 = 1;}
20)         W_candidate = W_candidate -{w_*};
21)         **If** W_candidate = {}: b_2 = 1;
22)     **If** (b_3 = = 1): //Recruit the qualified neighbor
23)         W_B = W_B ∪{w_*};
24)         Lacking_S_B = Lacking_S_B-$S_{w*}$;
25)     **If** (Lacking_S_B = = {}): b_1 = 1;
26)     **If** (W_temp2 = = W) **and** (W_candidate = = {}): b_1 = 1;
27) **Return** (W_B);
28) **End**.
```

Definition 4. *Local crowdsourcing value of a worker for one task.* Let there be an existing worker team W_{w_x} for task t_x; $W_{w_x} \subseteq W$. The local crowdsourcing value of w_i perceived by W_{w_x} for t_x is defined as:

$$LV_{w_i}(t_x) = \frac{\alpha_1 \cdot (|(S_{t_x} - \cup_{\forall w_j \in W_{t_x}} S_{w_j}) \cap S_{w_i}|/|S_{t_x}|) + \alpha_4 \cdot R_{w_i}}{\alpha_2 \cdot D^{team}_{w_i}(t_x) + \alpha_3 \cdot (\gamma_{w_i}/b_{t_x})} \tag{31}$$

where $\boldsymbol{\alpha}_1 + \boldsymbol{\alpha}_2 + \boldsymbol{\alpha}_3 + \boldsymbol{\alpha}_4 = 1$, *and*

$$D^{team}_{w_i}(t_x) = \frac{1}{|W_{t_x}|} \sum_{\forall w_j \in W_{t_x}} (d_{ij}/(|S_{t_x} \cap S_{w_j}|/|S_{t_x}|)) \tag{32}$$

5.2 Distributed Formation of the Basic Team

5.2.1 Basic Set of Skills for a Batch of Tasks

We can use three methods to decide the basic set of skills for a batch of tasks.

- Adopting the skills of the first task if the tasks in a batch need to be executed in sequence. Let $B = \{t_1, t_2, \ldots, t_n\}$. $\forall t_x \wedge (1 \le x \le n\text{-}1)$, it is assumed that t_x is executed before t_{x+1}; then, the basic set of skills of B is $Basic\_S_B = S_{t_1}$.
- Adopting the skills of the core task if the tasks in a batch do not need to be executed in sequence. The core task is defined as the task with the minimum sum of distances with other tasks in B:

$$t_c = \arg\min_{\forall t_x \in B} \left(\sum_{\forall t_y \in (B - \{t_x\})} \delta_{x,y} \right) \tag{33}$$

Now the basic set of skills of B is $Basic\_S_B = S_{t_c}$.

- Adopting the intersection of the skills of all tasks in B, i.e., $Basic\_S_B = \cap_{\forall tx \in B} S_{t_x}$. This method can avoid the costs of eliminating unutilized workers.

5.2.2 Forming the Basic Team

We construct a virtual basic task, t_v, which only requires the skills of $Basic\_S_B$. Similar to Algorithm 1, we will form a team for t_v. This team formation process can use Algorithm 1 by making certain revisions. The set of skills is the basic set of skills but not the set of all skills required by the batch of tasks. Revising the calculations of crowdsourcing values according to Definition 3 and 4 is required. Moreover, the calculation of real payment can be revised by only considering the individual virtual basic task but not the batch of tasks. The process can be shown as the algorithm in the Appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2022.3161019. Finally, the original basic team for t_v can be achieved, denoted as W_{t_v} .

Because the basic team is crucial, we here refine the above team formation method. In Algorithm 1, the existing team members will remain fixed during the whole formation process; such method may only form the local optimal team, but some better workers may be not recruited by the team. To address this problem, after the original team is formed by using the method similar to Algorithm 1, we can add a refinement process. Now, a worker with the maximum local crowdsourcing value is selected from the neighbors of all existing team members; the team can attempt to replace each team member by the selected worker. If the performance of the new team is better than that of the previous team, the new team can replace the previous team. This process will repeat until no workers can be found to replace the existing team members.

To consider the optimization objective, we can use the following index to measure the performance of a team:

$$Per(W_{t_v}) = \alpha_1 \cdot |W_{t_v}| + \alpha_2 \cdot \sum_{\forall w_i \in W_{t_v}} \gamma_{w_i} + \alpha_3 \cdot \frac{1}{2} \sum_{\forall w_i, w_j \in W_{t_v}} d_{ij} + \alpha_4 \cdot \sum_{\forall w_i \in W_{t_v}} \frac{1}{R_{w_i}} \tag{34}$$

The refinement process is shown as Algorithm 2. Algorithm 2's time complexity is O(|W|^2), which is more complex than Algorithm 1 whose time complexity is O(|S_B|·|W|), because in reality |W| is much higher than |S_B|. This can explain why we do not use the refinement in Algorithm 1 because we want to save time complexity.

Algorithm 2. Refinement of a team (W_{t_v}). /* W_{t_v} is the original basic team for virtual task t_v; W is the set of workers in the social network */

```
1)  $W_{temp1} = W_{t_v}$; $b_1 = 0$;
2)  While ($b_1 = = 0$) do: //Refine the team
3)    $W_{neighbor} = \{\}$; //Neighbors of observed workers
4)    $\forall w_i \in W_{temp1}$:
5)      $\forall w_j \in Neigh(w_i)$: //Neighbors of $w_i$
6)        If $w_j \notin W_{temp1}$: $W_{neighbor} = W_{neighbor} \cup \{w_j\}$;
7)    If ($W_{neighbor} = \{\}$): $b_1 = 1$;//All workers are considered
8)    $W_{temp1} = W_{temp1} \cup W_{neighbor}$;
9)    While $W_{neighbor} \neq \{\}$ do:
10)     $w_* = \arg\max_{\forall w_i \in W_{neigbhor}}(LV_{w_i}(t_v))$;
11)     $W_{temp2} = W_{t_v}$ ;
12)     $\forall w_i \in W_{t_v}$:
13)       $W_{temp3} = (W_{t_v}$-$\{w_i\}) \cup \{w_*\}$;
14)       If $Res(w_*) \leq \delta$ and $time_{w_*}(t_v) \leq d_{t_v}$ and $S_{t_v} \subseteq \cup_{\forall w_i \in W_{temp3}} S_{w_i}$ and ($Per(W_{temp3}) < Per(W_{temp2})$):
15)         $W_{temp2} = W_{temp3}$;
16)     $W_{t_v} = W_{temp2}$;
17)     $W_{neighbor} = W_{neighbor}$-$\{w_*\}$;
18) Return ($W_{t_v}$);
19) End.
```

Theorem 2 ensures that the optimization objective in Eq. (12) can be approached by Algorithm 1. Now the basic team is formed by Algorithm 1 and the refinement process in Algorithm 2. Because the refinement process can achieve better results than Algorithm 1, our approach can ensure that the resulted basic team can achieve good performance.

5.3 Dynamic Adjustment of the Team

5.3.1 Execution Sequence of Tasks

There are two methods for determining the execution sequence of tasks.

In the first method, the execution sequence of the tasks is predefined, i.e., given a batch of tasks, $B = \{t_1, t_2, \ldots, t_n\}$, $\forall t_x \wedge (1 \leq x \leq n\text{-}1)$, t_x is executed before t_{x+1}. If we use an array, $T[x]$, to denote the execution sequence, we have $T[x] = t_x$.

Algorithm 3. Determination of the execution sequence of a batch of tasks.

```
1) $x = 0$; $B_{temp} = B$; $Min\_dis$ = max value; task $T[|B|]$;
2) While $B_{temp} \neq \{\}$ do:
3)     $\forall t_y \in B_{temp}$:
4)       If $\delta_{y,c} < Min\_dis$:
5)         $\{t_{temp} = t_y$; $Min\_dis = \delta_{y,c}$;$\}$
6)   $x{+}{+}$; $T[x] = t_{temp}$;
7)   $B_{temp} = B_{temp}$-$\{t_{temp}\}$; $Min\_dis = +\infty$;
8) Return ($T[x]$);
9) End.
```

In the second method, the execution order of every task in the batch is determined by its skill distance to the virtual basic task. This idea is practical because the method can make it easier for the execution of one task to utilize the existing execution results of other similar finished tasks. Let the virtual basic task of a batch of task B be t_v. The algorithm for determining the execution sequence of tasks in B is shown as Algorithm 3, which is O($|B|^2$).

5.3.2 Dynamic Adjustment

The basic team will be adjusted for each task in the batch by eliminating some existing members and recruiting new members. While an existing team is assigned a new task in the batch, the unutilized team member who cannot provide any skills required by the new task will be eliminated. The remaining team members will recruit new members to satisfy the skill requirement of the new task by observing the local crowdsourcing values of other workers from the near to the distant in the social network.

Algorithm 4. Dynamic team adjustment for each task in a batch.

```
1)  For (x = 1;x< = |B|) do: //Team adjusting for each task
2)    $t_x = T[x]$; $W_{t_x} = W_{t_v}$ ;
3)    For all $w_i \in W_{t_x}$: //Eliminate the useless members
4)      If ($S_{w_i} \cap S_{t_x} = = \{\}$): $W_{t_x} = W_{t_x}$ -$\{w_i\}$;
5)    $b_1 = 0$; $S_{t_x-lacking} = S_{t_x} - \cup_{\forall w_i \in W_{t_x}} S_{w_i}$;
6)    If ( $S_{t_x-lacking} = = \{\}$): $b_1 = 1$;
7)    While ($b_1 = = 0$) do: //Recruit new members
8)      $W_{neighbor} = \{\}$; //Neighbors of the members
9)        $\forall w_i \in W_{t_x}$:
10)         $\forall w_j \in Neigh(w_x)$: //Neighbors of $w_i$
11)           If $w_j \notin W_{t_x}$: $W_{neighbor} = W_{neighbor} \cup \{w_j\}$;
12)   If ($W_{neighbor} = = \{\}$): $b_1 = 1$;
13)   $b_2 = 0$; $b_3 = 0$;
14)   While ($b_2 = = 0$) do:
16)     $w_* = \arg\max_{\forall w_i \in W_{neigbhor}}(LV_{w_i}(t_x))$
17)       If $Res(w_*) \leq \delta$ and $time_{w_*}(t_x) \leq d_{t_x}$
          and ($P_{w_*}(t_x) \geq \gamma_{w_*}$)and ($S_{w*} \cap S_{t_x-lacking} \neq \{\}$):
18)       $\{b_2 = 1; b_3 = 1;\}$
19)     $W_{neighbor} = W_{neighbor}$-$\{w_*\}$;
20)     If ($W_{neighbor} = = \{\}$): $b_2 = 1$;
21)   If ($b_3 = = 1$):
22)     $W_{t_x} = W_{t_x} \cup \{w_*\}$;
23)     $S_{t_x-lacking} = S_{t_x-lacking}$ -$S_{w*}$;
24)   If ( $S_{t_x-lacking} = = \{\}$): $b_1 = 1$
25) Output ($W_{t_x}$);
26) End.
```

Let the virtual basic task of batch B be t_v. If the basic team for B is W_{t_v}, the dynamic adjustment of the team for each task in B is shown as Algorithm 4, which is O($|B| \cdot |W|^2$). W_{t_x} denotes the final team for task t_x.

Theorem 3. Let the basic team for a batch of tasks B be W_{t_v} . For any task in B, $t_x(t_x \in B)$, the final team after adjustment using Algorithm 4 is W_{t_x}. It is then assumed that there is another worker team W_{t_x}', which is formed by eliminating the members in W_{t_v} that cannot provide any skills for t_x and randomly recruiting workers whose reservation wages can be satisfied. The skills of all members in W_{t_x}' can fully satisfy the skill requirements of t_x. Thus, we have

$$(\alpha_1 \cdot C_{Form}(B)\_W_{t_x} + \alpha_2 \cdot C_{Pay}(B)\_W_{t_x} + \alpha_3 \cdot C_{Com}(B)\_W_{t_x} + \alpha_4 / R\_W_{t_x})$$
$$\leq \left(\alpha_1 \cdot C_{Form}(B)\_W'_{t_x} + \alpha_2 \cdot C_{Pay}(B)\_W'_{t_x} + \alpha_3 \cdot C_{Com}(B)\_W'_{t_x} + \alpha_4 / R\_W'_{t_x}\right)$$

Proof. We can use reductio ad absurdum to prove Theorem 3. Let the set of remaining team members of W_{t_v} by eliminating the members who cannot provide any skills for t_x be $Net\_W_{t_x}$. Assume there is a set of workers $W_{t_x}{}'$, $W_{t_x}{}' \neq W_{t_x}$, formed by $Net\_W_{t_x}$ randomly recruiting some workers whose reservation wages can be satisfied, and $\Sigma_{\forall w_i \in W'_{t_x}} S_{w_i} \supseteq S_{t_x}$. If the assumption

$$\begin{aligned}&(\alpha_1 \cdot C_{Form}(B)\_W_{t_x} + \alpha_2 \cdot C_{Pay}(B)\_W_{t_x} + \alpha_3 \cdot C_{Com}(B)\_W_{t_x} + \alpha_4/R\_W_{t_x})\\ &> \left(\alpha_1 \cdot C_{Form}(B)\_W'_{t_x} + \alpha_2 \cdot C_{Pay}(B)\_W'_{t_x} + \alpha_3 \cdot C_{Com}(B)\_W'_{t_x} + \alpha_4/R\_W'_{t_x}\right)\end{aligned}$$

is true, then there exists at least one worker with a higher local crowdsourcing value perceived by $Net\_W_{t_x}$ for t_x and whose reservation wage can be satisfied (and can provide any lacking skills for t_x) but who cannot be selected by $Net\_W_{t_x}$ using Algorithm 4, and another worker with a lower local crowdsourcing value perceived by $Net\_W_{t_x}$ is selected. However, from Step 16 in Algorithm 4, in each round for recruiting new member, the worker with the highest local crowdsourcing value whose reservation wage can be satisfied will be the first to be definitely selected by $Net\_W_{t_x}$ if the worker can provide any currently lacking skills for t_x. Therefore, the above assumption cannot occur in reality when Algorithm 4 is used. □

Theorem 3 ensures that the optimization objective in Eq. (17) can be approached by Algorithm 4.

6 Experiments

6.1 Benchmark Approaches

Because previous related studies always adopt the individual task-specific team formation approach [4], [5], [6], [7], [8], our approaches, distributed formation approach of a fixed team for batch crowdsourcing (*Our algorithm-fixed*) and distributed formation approach of a dynamic team for batch crowdsourcing (*Our algorithm-dynamic*) are compared to this benchmark approach. Moreover, the greedy algorithm is an often used optimization heuristic for problem solving, thus it can be used as another benchmark for comparison. Therefore, we also compare our approaches with two greedy algorithms: centralized greedy algorithm which achieves a fixed team, and distributed greedy algorithm which achieves dynamic teams.

- Individual task-specific team formation (*Individual algorithm*). For each task, a new team is formed from scratch. For each team, all members are selected by the requester centrally according to their crowdsourcing values.
- *Centralized greedy algorithm*. In this algorithm, a fixed team for all tasks in a batch is formed centrally using the greedy method. At each step, the system will select a new team member to achieve the locally optimal result for satisfying the optimization objective. Finally, a team can be achieved to satisfy the requirements of all tasks in the batch.
- *Distributed greedy algorithm*. In this algorithm, dynamic teams are formed for a batch of tasks, i.e., each task has a different team. In the formation of each team, first, the system randomly selects a worker as the initiator for forming the team. The existing team members will select new members greedily from their neighbors, i.e., the neighbor who can achieve the locally optimal result is recruited to join the team. The members of the new team will repeat this recruiting process to select other workers to enlarge the team until the skills of all tasks in the batch can be satisfied or all workers in the crowd are observed.

6.2 Dataset and Data Processing

The experiment dataset was collected from the "web-mobile-software-development" category at the Upwork between July 31st and August 6th, 2021 (The reason why we only collected the data of tasks within one week is that most tasks at the Upwork can be assigned successfully within one week; and the reason why we only used the tasks within the same category is that the same category of tasks are often similar and can be addressed in batch). The amounts of collected workers and tasks are 6120 and 7651 respectively. The workers in the experiments are interconnected by three typical social network structures: small-world networks, scale-free networks, and random networkers. In the experiments, the estimated completion time of a task in the batch is the maximum completion time of all workers in the team for completing the task.

The maximum response time in the experiments is predefined according to the statistics on the response time at the real-world crowdsourcing website. After counting the response time at Upwork website, we find that most tasks may receive 5-10 proposals after the tasks are published for 40 minutes at the website. Because less than 5 proposals may lead to inaccurate results, the response time of 40 minutes can be accepted. Therefore, we set the maximum response time to 40 minutes.

In the experiments, the size of batch can be set for varying cases. With a predefined size of batch, let n, we can select the n most similar tasks to form a batch; after the first batch is formed, we will select the n most similar tasks from the remaining tasks to form the second batch; such process will be repeated until all tasks are formed into any batches.

6.3 Performance Indices

According to the optimization objective of this paper, we define the following indices to evaluate the performance of each approach.

- Cost of Team Formation: According to Eq. (6), we assume that the cost of team formation is generated by the process of finding team members in the social network.
- Cost of Communication: After the team is formed, the team members will communicate with each other to cooperate. The communication cost between two workers is measured by the hops of the shortest path between them in the social network. Each worker broadcasts the message to his/her immediate neighbors in the social network if he/she wants to coordinate with other workers.
- Total Payment by Requester: According to Eqs. (7) and (8), this is a discount of payment to workers, so we define this index to evaluate our optimization objective-minimizing requesters' real payment.
- Success Rate of Tasks: This index is the ratio of number of tasks completed successfully to the number

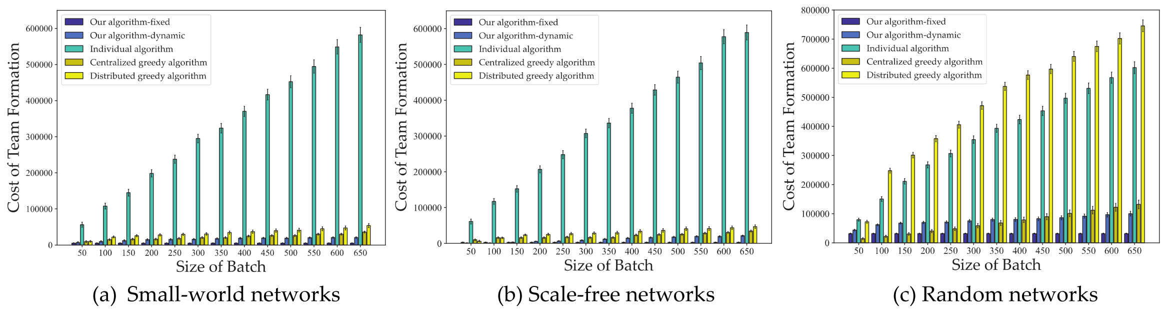


Fig. 4. Tests on the cost of team formation.

of all outsourced tasks, which can measure the effectiveness of the reputation mechanism in our approaches and the crowdsourcing quality.

6.4 Experimental Results

Our experiments are implemented by Java and under Intel (R) Core(TM) CPU i7-6700 3.4GHz and 8G memory. Each experiment is finished until the system reaches on a steady state and is repeated 20 times; the final result is obtained from the average of the 20 times of results. Without loss of generality, this paper adopts the 95% confidence level. The experimental results are shown in the figures by reporting error bars. The error bar means the confidence interval giving an idea of the quality of the estimated mean value, which is denoted as a range of values of tested performance indices with a certain degree of confidence in the experiments.

6.4.1 Tests on Cost of Team Formation

Now we test the performances on the cost of team formation, shown in Fig. 4. We can see that both of our approaches can result in lower costs of team formation. Moreover, while the size of batch becomes larger, the costs of team formation of our two approaches have no obvious changes. In comparison, the cost of team formation of *Our algorithm-fixed* is less than that of *Our algorithm-dynamic* in small networks and random networks; therefore, it denotes that the dynamic adjustment of team will produce higher costs in these two types of networks.

We find that the *Individual algorithm* produces higher cost for team formation because each task needs to form a new team. The cost of team formation of the *Centralized greedy algorithm* is less than that of the *Distributed greedy algorithm* because the former algorithm only needs to form a fixed team for all tasks in the batch, but the latter algorithm needs to form dynamic teams for tasks in the batch. Especially in Fig. 4(c), the performance of the cost of team formation of the *Batch-distributed-greedy algorithm* is higher than that of the *Individual algorithm* even when the size of the batch becomes larger. The potential reason is that the random network structure may make the former algorithm difficult to find new team members from the neighbors of existing team members.

In summary, our two approaches perform better than previous benchmark approaches in terms of the cost of team formation; moreover, our two approaches have good scalability for the sizes of the batches.

6.4.2 Tests on Cost of Communication

We now test the performances on the cost of communication, shown in Fig. 5. We can see that our two presented approaches result in lower communication cost. In the small-world and random networks, *Our algorithm-fixed* performs better than *Our algorithm-dynamic*; but in the scale-free networks, the two presented approaches perform similarly. Therefore, it shows that the scale-free networks can provide shorter communication lengths for both fixed team and dynamic teams. Moreover, the cost of communication performance of our two approaches has no obvious fluctuations with increasing batch size.

Among the three benchmark approaches, the *Centralized greedy algorithm* performs the worst. The reason is that the team members are selected from the whole network by considering the optimization objective but overlooking the communication distances among team members.

In summary, our two approaches perform better than previous benchmark approaches in terms of the cost of communication; moreover, our two approaches have good scalability for the sizes of the batches.

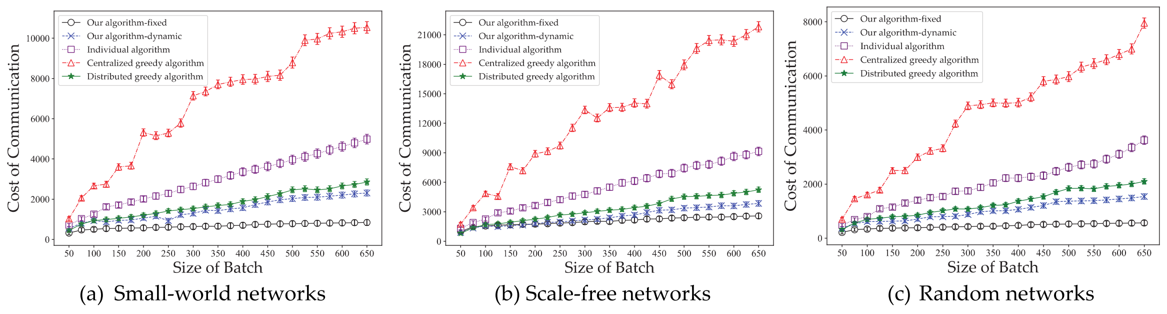
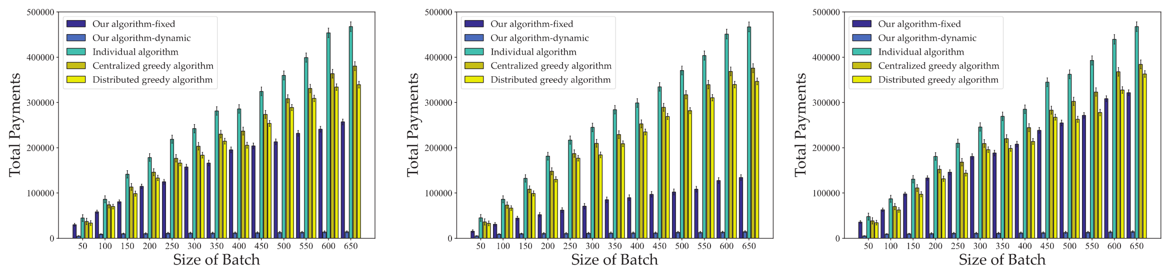


Fig. 5. Tests on the cost of communication.



(a) Small-world networks (b) Scale-free networks (c) Random networks

Fig. 6. Tests on the total payments by requesters.

6.4.3 Tests on Total Payment By Requesters

We now test the total payments by requesters, shown in Fig. 6. We can see that our two presented approaches achieve lower total payments. The reason is that our approaches adopt the discounting mechanism, but the previous approaches did not. Moreover, we can see that *Our algorithm-fixed* may produce higher total payments by requesters than *Our algorithm-dynamic*; the reason is that each requester will pay all team members, even the members who do not factually execute the task in the former approach, but the requester will only pay the team members that factually execute the task in the latter approach.

Moreover, with the increase of size of batch, the total payments by the requesters of our two approaches have no obvious change, but those of other three approaches can increase with the increasing size of the batch. Therefore, our presented two approaches have good scalability in the performance of total payments.

6.4.4 Tests on Success Rate of Tasks

We allow some workers to fabricate their skills with certain probabilities. If an assigned worker can execute the task successfully, his reputation will become higher, and vice versa. Now we make a series of experiments on the success rate of tasks, in which the subsequent experiments can utilize the reputations of workers achieved by the previous experiments. The results are shown in Fig. 7.

We can see that our two presented approaches have higher success rate in all experiments under the three typical network structures; moreover, with the progress of experiments, the advantages of our approaches become more obvious. Therefore, the reputation mechanism in our approaches can effectively improve the success rate of tasks. Moreover, we can see that *Our algorithm-dynamic* may produce lightly higher success rate of tasks than *Our algorithm-fixed*; the reason is that the refinement algorithm can form a better team with higher reputations.

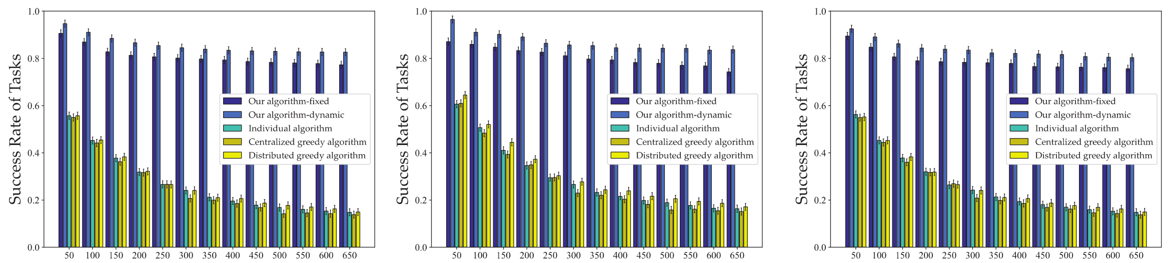
6.4.5 Tests on the Generality Under Varying Amounts of Tasks and Workers

At first we randomly select 6000 tasks and 5000 workers from the initial 7651 tasks and 6120 workers collected from the "web-mobile-software-development" category at the Upwork between July 31st and August 6th, 2021; then, we expand the set of tasks by introducing more tasks newly collected from the same category. The amounts of tasks vary from 6000 to 11000, and the amounts of workers vary from 5000 to 10000, in the experiments. The generality experimental results are shown in Figs. 8 and 9. We can see that our approaches always outperform the other three approaches in terms of the four performance indices under varying amounts of tasks and workers, which shows the generality of the proposed approaches.

7 Discussion and Critical Reflection

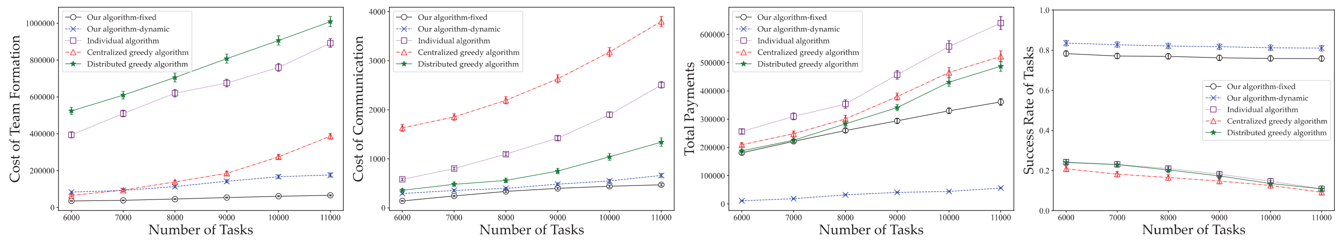
This paper mainly considers the technological aspects of the problem. Here we make discussion and critical reflection by considering more human and social factors.

- In fact there may be some human factors that may influence the selection of new teammates, such as acquaintance and cooperation history. To consider these human factors in our approach, we can combine the quantified human factors (such as acquaintance degree, cooperation history, etc.) into the definition of the local crowdsourcing value of a worker and set weights between the human factors and the technological factors. Therefore, our approach can easily be extended to consider the human factors. Certainly, how to measure and quantify



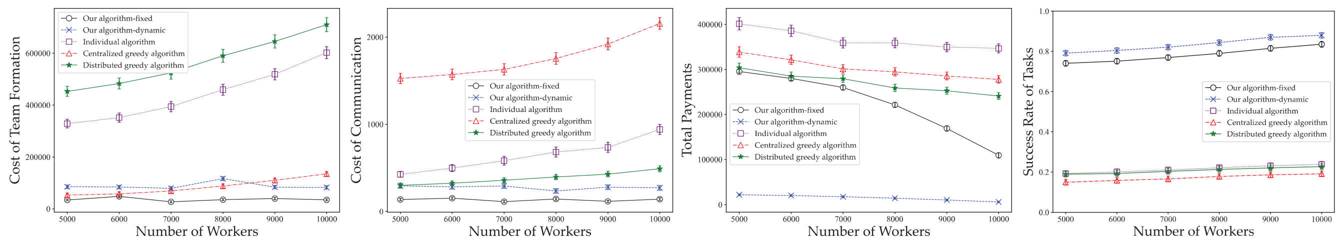
(a) Small-world networks (b) Scale-free networks (c) Random networks

Fig. 7. Tests on the success rate of tasks.



(a) Cost of Team Formation (b) Cost of Communication (c) Total Payments by requests (d) Success Rate of Tasks

Fig. 8. Tests on the generality under varying amounts of tasks.



(a) Cost of Team Formation (b) Cost of Communication (c) Total Payments by requests (d) Success Rate of Tasks

Fig. 9. Tests on the generality under varying amounts of workers.

the human factor and combine them with the technological factors uniformly is our future work.

- With our approach, the assigned workers can achieve more payments and be increased higher reputations, which will in return help them to be assigned more tasks in the future. However, new workers may not be recruited by any teams since they have no reputations. Therefore, our approach may increase the imbalance between good-earning and low-paid workers, and what is more serious is that some senior workers may monopolize the crowdsourcing markets. Obviously, such trend may violate the original intention of crowdsourcing that try to utilize the abilities of the generally large group of people. To address such problem, the new workers can be endowed with initial reputations and load balancing measures can be adopted in the future work.

8 Conclusion

Previous team formation-based crowdsourcing studies may be costly and cannot apply to crowdsourcing markets where the number of tasks is large because each team is tailored only for one task; moreover, the centralized manner used in these studies may place a heavy burden on requesters. Therefore, this paper presents a batch crowdsourcing approach based on distributed team formation in which tasks with similar skill requirements can be addressed in a batch to save computational cost and where workers can self-organize the team formation though their social networks.

We formalize the optimization objective of the problem and prove that it is NP-hard; then, we present two types of heuristic approaches. The experiments on a real-world dataset show that our two presented approaches have better performance in terms of the three typical types of cost in team formation and the success ratio of tasks by comparing them with previous benchmark approaches.

In the future, we will explore the adaption of the distributed team formation to dynamic environments in which available workers may join or depart dynamically, and their social networks may change dynamically.

Acknowledgments

Jiuchuan Jiang and Kai Di are the co-corresponding authors.

References

[1] L. Y. Tong *et al.*, "SLADE: A smart large-scale task decomposer in crowdsourcing," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 8, pp. 1588–1591, Jan. 2018.

[2] E. Estellés-Arolas and F. González-Ladrón-de-Guevara, "Towards an integrated crowdsourcing definition," *J. Inf. Sci.*, vol. 38, no. 2, pp. 189–200, Apr. 2012.

[3] V. Rajan *et al.*, "Crowd control: An online learning approach for optimal task scheduling in a dynamic crowd platform," in *Proc. ICML Workshop Mach. Learn. Meets Crowdsourcing*, Atlanta, GA, vol. 2, Jun. 2013.

[4] I. Lykourentzou, R. E. Kraut, S. Wang, and S. P. Dow, "Team dating: A self-organized team formation strategy for collaborative crowdsourcing," in *Proc. ACM Conf. Hum. Factors Comput. Syst*, San Jose, CA, USA, May 2016, pp. 1243–1249.

[5] Z. Pan, H. Yu, C. Miao, and C. Leung, "Efficient collaborative crowdsourcing," in *Proc. 30th AAAI Conf. Artif. Intell.*, Phoenix, AZ, USA, Feb. 2016, pp. 4248–4249.

[6] W. Wang, J. Jiang, B. An, and Y. Jiang, "Towards efficient team formation for crowdsourcing in non-cooperative social networks," *IEEE Trans. Cybern.*, vol. 47, no. 12, pp. 4208–4222, Sep. 2017.

[7] Q. Liu, T. Luo, and R. Tang. S. Bressan, "An efficient and truthful pricing mechanism for team formation in crowdsourcing markets," in *Proc. IEEE Int. Conf. Commun.*, London, U.K., Jun. 2015, pp. 567–572.

[8] M. Fathian, M. Saei-Shahi, and A. Makui, "A new optimization model for reliable team formation problem considering experts' collaboration network," *IEEE Trans. Eng. Manage.*, vol. 64, no. 4, pp. 586–593, Jul. 2017.

[9] M. Yin and M. L. Gray, "The communication network within the crowd," in *Proc. 25th Int. World Wide Web Conf.*, Montreal, Canada, Apr. 2016, pp. 1293–1303.

[10] M. L. Gray and S. Suri, "The crowd is a collaborative network," in *Proc. 19th ACM Conf. Comput.-Supported Cooperative Work Social Comput.*, San Francisco, USA, Feb./Mar. 2016, pp. 134–147.

[11] Y. Jiang, "A survey of task allocation and load balancing in distributed systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 2, pp. 585–599, Feb. 2016.

[12] V. Raychoudhury, S. Shrivastav, S. S. Sandha, and J. Cao, "CROWD-PAN-360: Crowdsourcing based context-aware panoramic map generation for smartphone users," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 8, pp. 2208–2219, Aug. 2015.

[13] D. P. Forbes, P. S. Borchert, M. E. Zellmer-Bruhn, and H. J. Sapienza, "Entrepreneurial team formation: An exploration of new member addition," *Entrepreneurship Theory Pract.*, vol. 30, no. 2, pp. 225–248, Mar. 2006.

[14] T. Lappas, K. Liu, and E. Terzi, "Finding a team of experts in social networks," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, Paris, France, Jun./Jul. 2009, pp. 127–132.

[15] M. Kargar, A. An, and M. Zihayat, "Efficient Bi-objective team formation in social networks," in *Proc. Eur. Conf. Mach. Learn. Princ. Pract. Knowl. Discov. Databases*, Bristol, U.K., Sep. 2012, pp. 483–498.

[16] L. Tran-Thanh, T. D. Huynh, A. Rosenfeld, S. Ramchurn, and N. R. Jennings, "BudgetFix: Budget limited crowdsourcing for interdependent task allocation with quality guarantees," in *Proc. 13th Int. Conf. Auton. Agents Multiagent Syst.*, Paris, France, May 2014, pp. 477–484.

[17] A. Kittur *et al.*, "CrowdWeaver: Visually managing complex crowd work," in *Proc. ACM Conf. Comput. Supported Cooperative Work*, Seattle, WA, USA, Feb. 2012, pp. 1033–1036.

[18] A. Xu, H. Rao, S. P. Dow, and B. P. Bailey, "A classroom study of using crowd feedback in the iterative design process," in *Proc. 18th ACM Conf. Comput. Supported Cooperative Work Social Comput.*, Vancouver, Canada, Mar. 2015, pp. 1637–1648.

[19] F. Li, Y. Ding, M. Zhou, K. Hao, and L. Chen, "An affection-based dynamic leader selection model for formation control in multirobot systems," *IEEE Trans. Syst., Man, Cybern.: Syst.*, vol. 47, no. 7, pp. 1217–1228, Aug. 2017.

[20] Y. Zhang and M. van der. Schaar, "Reputation-Based incentive protocols in crowdsourcing applications," in *Proc. 31rd Annu. IEEE Int. Conf. Comput. Commun.*, Orlando, USA, Mar. 2012, pp. 2140–2148.

[21] C. Schwartz *et al.*, "Modeling crowdsourcing platforms to enable workforce dimensioning," in *Proc. Int. Telecommun. Netw. Appl. Conf.*, Sydney, Australia, Nov. 2015, pp. 30–37.

[22] M. Rokicki, S. Zerr, and S. Siersdorfer, "Groupsourcing: Team competition designs for crowdsourcing," in *Proc. 24th Int. Conf. World Wide Web*, Florence, Italy, May 2015, pp. 906–915.

[23] S. J. Chen and L. Lin, "Modeling team member characteristics for the formation of a multifunctional team in concurrent engineering," *IEEE Trans. Eng. Manage.*, vol. 51, no. 2, pp. 111–124, Apr. 2004.

[24] S. Bhattacharya *et al.*, "Crowds of crowds: Performance-based modeling and optimization over multiple crowdsourcing platforms," *Hum. Computation*, vol. 2, no. 1, pp. 105–131, Aug. 2015.

[25] P. K. Kopalle, C. F. Mela, and L. Marsh, "The dynamic effect of discounting on sales: Empirical analysis and normative pricing implications," *Marketing Sci.*, vol. 18, no. 3, pp. 317–332, Aug. 1999.

[26] X. Chen, B. Proulx, X. Gong, and J. Zhang, "Exploiting social ties for cooperative D2D communications: A mobile social networking case," *IEEE/ACM Trans. Netw.*, vol. 23, no. 5, pp. 1471–1484, Jun. 2015.

[27] A. Kulkarni, M. Can, and B. Hartmann, "Collaboratively crowdsourcing workflows with turkomatic," in *Proc. ACM Conf. Comput. Supported Cooperative Work*, Seattle, WA, USA, Feb. 2012, pp. 1003–1012.

[28] M. Kargar and A. An, "Discovering top-k teams of experts with/without a leader in social networks," in *Proc. 20th ACM Int. Conf. Inf. Knowl. Manage.*, Glasgow, Scotland, U.K., Oct. 2011, pp. 985–994.

[29] B. Wang *et al.*, "On batch-processing based coded computing for heterogeneous distributed computing systems," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 3, pp. 2438–2454, Jul. 2021, doi: 10.1109/TNSE.2021.3095040.

[30] T. Alam and Z. Raza, "Batch scheduling model for distributed systems," in *Proc. 4th Int. Conf. Parallel, Distrib. Grid Comput.*, Waknaghat, India, Dec. 2016, pp. 79–83.

[31] P. Liu and Z. Li, "Task complexity: A review and conceptualization framework," *Int. J. Ind. Ergonom.*, vol. 42, no. 6, pp. 553–568, Nov. 2012.

[32] V. Freitas *et al.*, "A batch task migration approach for decentralized global rescheduling," in *Proc. 30th Int. Symp. Comput. Architecture High Perform. Comput.*, Lyon, France, Sep. 2018, pp. 49–56.

[33] J. Jiang, B. An, Y. Jiang, and D. Lin, "Context-aware reliable crowdsourcing in social networks," *IEEE Trans. Syst., Man, Cybern.: Syst.*, vol. 50, no. 2, pp. 617–632, Dec. 2020.

[34] R. T. Nakatsu, E. B. Grossman, and C. L. Iacovou, "A taxonomy of crowdsourcing based on task complexity," *J. Inf. Sci.*, vol. 40, no. 6, pp. 823–834, Dec. 2014.

[35] N. Nan and S. Kumar, "Joint effect of team structure and software architecture in open source software development," *IEEE Trans. Eng. Manage.*, vol. 60, no. 3, pp. 592–603, Jan. 2013.

[36] J. Hahn, J. Y. Moon, and C. Zhang, "Emergence of new project teams from open source software developer networks: Impact of prior collaboration ties," *Inf. Syst. Res.*, vol. 19, no. 3, pp. 369–391, Sep. 2008.

[37] J. Wang, S. Faridani, and P. G. Ipeirotis, "Estimating the completion time of crowdsourced tasks using survival analysis models," in *Proc. WSDM Workshop Crowdsourcing Search Data Mining*, Hong Kong, China, Feb. 2011, pp. 31–38.

[38] M. Allahbakhsh *et al.*, "Quality control in crowdsourcing systems," *IEEE Internet Comput.*, vol. 17, no. 2, pp. 76–81, Mar. 2013.

[39] J. Jiang *et al.*, "Understanding crowdsourcing systems from a multiagent perspective and approach," *ACM Trans. Auton. Adaptive Syst.*, vol. 13, no. 2, Jul. 2018, Art. no. 8.

[40] J. Jiang *et al.*, "Batch allocation for tasks with overlapping skill requirements in crowdsourcing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 8, pp. 1722–1737, Jan. 2019.

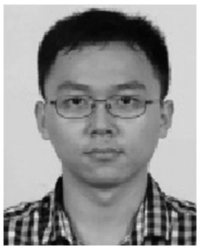
[41] S. Haefliger, G.V. Krogh, and S. Sapeth, "Code reuse in open source software," *Manage. Sci.*, vol. 54, no. 1, pp. 180–193, Jan. 2008.

Jiuchuan Jiang received the PhD degree in computer science from Nanyang Technological University, Singapore. He is currently an assistant professor with the School of Information Engineering, Nanjing University of Finance and Economics, Nanjing, China. He has authored or coauthored several scientific articles in refereed journals and conference proceedings, such as *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *ACM Transactions on Autonomous and Adaptive Systems*, and International Conference on Autonomous Agents and Multiagent Systems (AAMAS). His research interests include crowdsourcing, multiagent systems, and social networks.

Kai Di received the BS degree from the School of Information Science and Technology, Southwest Jiaotong University, Chengdu, China, in 2016. He is currently working toward the PhD degree with the School of Computer Science and Engineering, Southeast University, Nanjing, China. He has authored or coauthored several scientific articles in refereed journals and conference proceedings, such as *ACM Transactions on Autonomous and Adaptive Systems*, *ACM Transactions on Intelligent Systems and Technology*, and *IEEE International Conference on Systems, Man, and Cybernetics*. His current research focuses on multiagent systems.

Bo An (Member, IEEE) received the PhD degree in computer science from the University of Massachusetts, Amherst, USA. He is currently a president's council chair associate professor of computer science and engineering, and the co-director of Artificial Intelligence Research Institute, Nanyang Technological University, Singapore. He has authored or coauthored more than 100 referred papers at AAMAS, IJCAI, AAAI, ICAPS, KDD, UAI, EC, WWW, ICLR, NeurIPS, ICML, JAAMAS, AIJ and ACM/IEEE Transactions. His research interests include artificial intelligence, multiagent systems, computational game theory, reinforcement learning, and optimization.

Yichuan Jiang (Senior Member, IEEE) received the PhD degree in computer science from Fudan University, Shanghai, China, in 2005. He is currently a distinguished professor and the director of the Laboratory of Intelligent Systems and Social Computing, School of Computer Science and Engineering, Southeast University, Nanjing, China. He has authored or coauthored more than 100 scientific articles in refereed journals, such as the *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Journal on Selected Areas in Communications*, *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *IEEE Transactions on Cybernetics*, *ACM Transactions on Autonomous and Adaptive Systems*, *ACM Transaction on Intelligent Systems and Technology*, *Journal of Autonomous Agents and Multi-Agent Systems*, and in conference proceedings, such as IJCAI, AAAI, and AAMAS. His research interests include multiagent systems, social computing, and social networks.



Zhan Bu received the PhD degree in computer science from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2014. He is currently an associate professor with the College of Information Engineering, Nanjing University of Finance and Economics, Nanjing, China. His research interests include social networks and multiagent systems.

Jie Cao received the PhD degree from Southeast University, Nanjing, China, in 2002. He is currently a professor and the dean of the College of Information Engineering, Nanjing University of Finance and Economics, Nanjing, China. His research interests include business intelligence and social networks.