Toward Efficient Team Formation for Crowdsourcing in Noncooperative Social Networks

Wanyuan Wang, Jiuchuan Jiang, Bo An, Yichuan Jiang, *Senior Member, IEEE*, and Bing Chen

***Abstract*—Crowdsourcing has become a popular service computing paradigm for requesters to integrate the ubiquitous human-intelligence services for tasks that are difficult for computers but trivial for humans. This paper focuses on crowdsourcing complex tasks by team formation in social networks (SNs) where a requester connects to a large number of workers. A good indicator of efficient team collaboration is the social connection among workers. Most previous social team formation approaches, however, either assume that the requester can maintain information of all workers and can directly communicate with them to build teams, or assume that the workers are cooperative and be willing to join the specific team built by the requester, both of which are impractical in many real situations. To this end, this paper first models each worker as a selfish entity, where the requester prefers to hire inexpensive workers that require less payment and workers prefer to join the profitable teams where they can gain high revenue. Within the noncooperative SNs, a distributed negotiation-based team formation mechanism is designed for the requester to decide which worker to hire and for the worker to decide which team to join and how much should be paid for his skill service provision. The proposed social team formation approach can always build collaborative teams by allowing team members to form a connected graph such that they can work together efficiently. Finally, we conduct a set of experiments on real dataset of workers to evaluate the effectiveness of our approach. The experimental results show that our approach can: 1) preserve considerable social welfare by comparing the benchmark centralized approaches and 2) form the profitable teams within less negotiation time by comparing the traditional distributed approaches, making our approach a more economic option for real-world applications.**

***Index Terms*—Crowdsourcing, multiagent, negotiation, noncooperative, social networks (SNs), team formation.**

Manuscript received February 3, 2016; revised June 3, 2016; accepted August 20, 2016. Date of publication September 7, 2016; date of current version November 15, 2017. This work was supported in part by the National Natural Science Foundation of China under Grant 61170164, Grant 61472079, and Grant 71201077, in part by the Funds for Distinguished Young Scholars of the Natural Science Foundation of Jiangsu Province under Grant BK2012020, and in part by the Collaborative Innovation Center of Novel Software Technology and Industrialization. Some preliminary results were presented and won the best student paper award in the 26th 2014 IEEE International Conference on Tools with Artificial Intelligence [27].This paper was recommended by Associate Editor J. Liu. *(Corresponding author: Yichuan Jiang.)*

W. Wang and Y. Jiang are with the School of Computer Science and Engineering, Southeast University, Nanjing 211189, China (e-mail: yjiang@seu.edu.cn).

J. Jiang and B. An are with the School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798.

B. Chen is with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China.

Digital Object Identifier 10.1109/TCYB.2016.2602498

I. Introduction

DRIVEN by the requirement of massive human intelligence service-oriented applications, such as data sensing [1], language translation [2], and image classification [3], crowdsourcing has become a flexible service computing paradigm by making the ubiquitous human resources available to requesters on demand [4]. By using certain crowdsourcing platform (e.g., Upwork, upwork.com), the requesters can advertise their tasks (i.e., service requirement) to workers and the workers choose whether or not to participate in and complete the tasks in change for monetary reward [5]. Typical crowdsourcing applications include solving simple tasks that each worker can complete each task independently and complex interdependent tasks that need to be solved in multiple phases [6] (e.g., sentence spelling mistakes correction) or teamwork [7], [8] (e.g., article editing in Wikipedia). In crowdsourcing environment, due to the diversity of workers' skills on performing tasks, previous researches mainly focus on whether the hired workers are professional enough such that they can satisfy a task's skill requirements [1]–[8].

In this paper, we concentrate on crowdsourcing an important class of complex tasks, where the success of completing such a complex task depends not only on the skills of the hired workers but also on how efficiently these workers can work together as a team [9]. For example, to develop a software product successfully, the product manager first needs to hire a group of professional software engineers with the necessary skills: *Requirement Analysis*, *Architecture Design*, *Implementation*, *Testing*, *Deployment*, and *Maintenance*. During software development, the engineer who performs *Testing* must communicate with the engineer who performs *Implementation* again and again to debug and optimize the software. Once the communication fails between the engineers due to language barriers or geographic distance, the product cannot be produced on time [10]. Therefore, the hired workers must also be able to cooperate with each other efficiently for team task completion. *Now the manager's problem is how to build such a professional and collaborative team of workers.*

With the advent of online social networks (SNs, e.g., LinkedIn, linkedin.com and GitHub, github.com), SNs provide good opportunities for the requester addressing this social team formation problem. On one hand, within SNs, the requester connects to a large number of social individuals and

can collect these individuals' public information (e.g., skill, salary requirement and working experience, etc.) by learning their profiles. This advantage will help the requester build professional teams [11], [12]. On the other hand, social connections among social individuals might represent collaboration relationships [13], [14] (e.g., collaborate on common task previously). The advantage of using these SNs is that the social individuals who have worked together previously are estimated to work effectively as a team without much coordination overhead [15], [16]. Motivated by this advantage, we consider building collaborative teams in SNs where team members form a connected graph such that they can work together efficiently [17]–[19].

Although a number of social team formation approaches have been proposed to build professional and collaborative teams in SNs [15], [16], [20]–[23], from which we find a couple of restrictive assumptions that we feel that are impractical. First, they assume that workers are cooperative, i.e., each worker is willing to join a team built by the requester to optimize the requester's objective. However, in practice, workers are always rational and their only incentive to join a team and provide skill services is to maximize their own benefits [24]–[26]. A practical social team formation should consider the worker's selfish nature. Second, they assume that the requester can maintain information of all workers and can directly communicate with them to build professional teams. However, because of privacy-preserving in SNs, the requester can only maintain limited information of its directly connected social neighbors [27]–[30]. Therefore, a practical social team formation should build teams in a natural distributed manner with partial information of socially-close partners.

To address these issues, this paper first models each individual as a selfish entity, where the requester aims to hire a collaborative team of inexpensive workers to minimize expense and the worker aims to join a profitable team and provide skill services to earn a considerable payment. Within the noncooperative SNs, we then present a distributed negotiation-based social team formation approach by allowing the requester to negotiate with the team members' neighbor workers only. As the main contribution of this paper, we rigorously design the negotiation mechanism for the requester and worker to make agreement on skill provision and payment for their own profit maximization. This negotiation-based social team formation mechanism mainly consists of the following three phases.

1) *Offer:* The requester interacts with the worker and sends an offer to the worker on skill provision and payment.
2) *Respond:* The worker responds (e.g., accept, reject, or propose skill payments improvement) to the requester on this offer.
3) *Confirm:* The requester makes the final confirmation on the received response and updates the team's profile.

Our theoretical analyses show that the requester can always find the most beneficial skills from the workers. Finally, we also conduct a set of experiments on real dataset to evaluate the proposed social team formation approach. The experimental results show that: 1) compared to the ideal centralized approach [17], [18], [22], our approach can preserve considerable social welfare (SW) and 2) compared to traditional distributed contract-net approach [31] and the greedy approach [28], our approach forms desirable teams by spending less budget and team formation time.

The remainder of this paper is organized as follows. In the next section, we provide a brief review of related work on crowdsourcing and team formation in SNs. In Section III, we give a formal definition of the social team formation problem. In Section IV, we present the framework of negotiation-based social team formation approach. In Section V, we discuss the negotiation mechanism employed by the requester and worker in detail. In Section VI, we conduct a set of experiments to evaluate the proposed approach's effectiveness. Finally, we give a conclusion of this paper and discuss the future work in Section VII.

II. Related Work

A. Crowdsourcing for Task Allocation

Crowdsourcing is a useful paradigm for requesters to harness the ubiquitous human resources on solving tasks that are difficult for computers. To motivate workers to contribute their personal resources, the workers should gain certain economic benefit from the task completion. However, because of the limited budget, the requester should decide how much should be paid to the workers. To address this issue, Azaria *et al.* [32] developed an agent-based reward determination mechanism by allowing agents to determine each worker's reward autonomously. To capture the dynamic characteristics of online labor markets, Singer and Mittal [33] introduced a truthful pricing mechanism which in addition is budget balanced. Unfortunately, due to the openness and dynamics of the online markets, workers might vary greatly in the quality of task completion [34]. Therefore, another key challenge in crowdsourcing is to guarantee the quality of task completion. To address this issue, a novel multiarmed bandit approach is devised for the requester to spend the first εB budget of its total budget B to derive workers' quality estimates and spend the remaining $(1-\varepsilon)B$ budget to maximize the task quality based on those estimates [34].

Although the above approaches are efficient in crowdsourcing simple tasks, there is a need to consider crowdsourcing a complex task that requires to be solved in multiple phases [6] or by teamwork [7], [8]. A popular method for solving a complex task is to decompose such a complex task into a flow of simple subtasks, allocate them to workers subsequently and finally combine the partial results of subtasks together to get the final solution [6]. These methods only focus on hiring professional workers [6]–[8]. In comparison, this paper not only crowdsources complex tasks by hiring professional workers but also guarantees the hired workers can collaborate with each other effectively. Being aware of the fact that teamwork can produce higher product quality than individuals alone, Rokicki *et al.* [35] allowed workers to build teams to improve the cost efficiency of crowdsourcing. On the other hand, Woolley *et al.* [36] empirically showed that a number of factors (e.g., social sensitivity and conversation democracy) are

correlated with team performance and Kittur and Kraut [8], Lykourentzou *et al.* [9], and Dow *et al.* [37] further verified that timely task-specific feedback, flexible coordination tools and personality similar can improve team performance. Complementary to these works [8], [9], [37], we aim to build the collaborative team with socially-close workers in SNs. Recently, being aware of the social influence and similar attributes among socially-close users, SNs have also been used in recommendation systems to improve the quality of recommendations [38]–[40]. To predict the rating of an item for a particular user, these methods [38]–[40] aggregate the ratings of a set of direct or indirect connected users on that item and then perform the recommendation based on the predicted ratings. These concepts (e.g., trust/influence propagation) proposed in the SN-based recommendation can inspire to form the collaborative team for SN-based crowdsourcing.

B. Team Formation in Social Networks

Wolf *et al.* [10] first realized that SNs play an important role in team collaboration. Meantime, Lappas *et al.* [15] first formally formulated the problem of team formation in SNs. They modeled each SN as a weighted and undirected graph $G = <V, E>$, where V represents individuals, E represents social connections, and the weights on the connections represent the communication cost between connected individuals. Given a social network G and a task J, Lappas *et al.* [15] target to find a team of individuals $V^* \subseteq V$ such that V^* not only meets all of the skill requirements of J but also incurs the least team communication cost. Following [15] are other social team formation variants with different application constraints and objectives, such as online dynamic settings where tasks arrive to the system dynamically [20]; expertise query settings where each required skill needs to be satisfied by adequate number of experts [21]; load balancing where the workers' assigned tasks should not exceed their capacity [16]; and budget minimization where the workers need to be paid for their skill contribution [22]. Rangapuram *et al.* [23] proposed a more generalized social team formation problem where team size, team budget, and team communication cost are considered. Noticing that all of these problems are NP-hard, the previous researches [15], [16], [20]–[23] focus on developing centralized approximations with rigorous quality guarantees. These centralizations, however, are impractical for real-world applications where many SNs do not involve such a central coordinator and individuals are always building or joining teams in a self-organized manner [25]–[28]. In comparison, this paper aims to design a distributed team formation approach for the requester to build teams and for workers to join teams.

Negotiation is a very effective distributed problem solving technique, which can be utilized to build social teams [41]–[45]. For example, Jiang *et al.* [41], [42] presented a contextual based team formation model, where the requester negotiates resources from nearby individuals to faraway individuals gradually. Recently, Wang and Jiang [43] proposed a game-theoretic team formation model by modeling each subtask as a cooperative mobile agent and then each agent targets to move to the individual that has the least workload. On the other hand, being aware of that the network has a dramatic effect on team performance [44], Gaston and desJardins [45] developed a dynamic structural adaption team formation method by allowing individuals to delete costly connections and rewire to professional individuals. Moreover, to improve team collaboration efficiency, system designer can facilitate social relationships to increase system social welfare [46], [47]. All of these proposals [41]–[48] assume that social individuals are cooperative, i.e., each individual is willing to join the specific team built by the requester. In contrast, this paper mainly focuses on building social teams in noncooperative SNs, where individuals are selfish with the aim of maximizing their own profit. Wang *et al.* [49] recently proposed a reverse auction-based truthful team formation in SNs that consider the individuals' selfish nature. This paper mainly focuses on maximizing social welfare (i.e., the aggregate utility of the requester and worker), while this paper attempts to maximize the requester and worker's utilities simultaneously.

De Weerdt *et al.* [31], Ye *et al.* [28], and Wang and Jiang [27] have made great effort on team formation in non-cooperative systems. For example, de Weerdt *et al.* [31] proposed a greedy neighbor-aware team formation model, where workers contribute their resources to the most desirable neighbor requester with the most profitable task. Although this model is easy to implement, its efficiency is limited due to its greedy nature. Ye *et al.* [28] considered a market-based team formation problem, where requesters have incomplete information (e.g., working cost) of workers. To address this issue, they proposed an efficient bilateral bargaining-based negotiation mechanism, where workers negotiate with each other until they reached an agreement on working cost. However, computing equilibrium strategies for such complex negotiation problem is difficult even impossible [26]. Thus, this paper proposes a set of negotiation strategies to compensate for these disadvantages. In addition, compared to [27], [28], and [31], our approach also considers forming social collaborative teams such that team members can work together efficiently.

III. Problem Description

In this section, we define the problem of team formation for crowdsourcing complex tasks in SNs. Table I gives a summary of notations used in this paper.

A. Social Networks

Each SN $= <A, E>$ is an unweight and undirected graph, where $A = \{a_1, a_2, \ldots, a_m\}$ is the set of agents (hereafter, we use the terms "agent," "worker," and "freelancer" interchangeably). $\forall(a_i, a_j) \in E$ represents the existence of a social connection between agents a_i and a_j. Those connections can provide high collaboration efficiency during task execution. Moreover, we assume that there are l type skills $S = \{s_1, s_2, \ldots, s_l\}$ available in an SN. We can transform any weight network to unweight network by removing the low social collaborations and regarding all of the remaining

TABLE I
DEFINITION OF NOTATION

Notation	Definition	Notation	Definition
a_i	Agent i	s_j	Skill j
$Q(a_i)$	The skills owned by agent a_i	$N(a_i)$	The neighbors of a_i
$c(a_i,s_j)$	The cost of a_i providing skill s_j	$p(a_i,s_j)$	The payment paid to a_i for providing skill s_j
κ	Task κ	Ia_κ	The initiator agent of task κ
R_κ	The skills required by task κ	It_κ	The initialization time of task κ
Dl_κ	The deadline of task κ	Wt_κ	The working time of task κ
$v_\kappa(t)$	The value function of task κ on time t	T_κ	The team built for task κ
Ω_κ	The team members of team T_κ	$Cont(\Omega_\kappa,\kappa,\cdot)$	The team members' skill contribution in T_κ
us_κ	The unsatisfied skills of team T_κ	$Ev(\kappa,t)$	The estimated value of task κ at time τ
$Ep(T_\kappa,\tau)$	The estimated profit of team T_κ at time τ	$Er(a_i,T_\kappa,\cdot)$	Agent a_i's estimated remuneration in T_κ
$Sr(T_\kappa)$	The success rate of forming a complete team for task κ	$Re(T_\kappa)$	The remuneration paid to team members in T_κ
$as(a_i,R_\kappa)$	Agent a_i's available skills for task κ, which is computed as $Q(a_i)\cap R_\kappa$	$Er(a_i,T_\kappa,ns,\cdot)$	Agent a_i's estimated remuneration in T_κ by providing skills ns

collaborations as the unweight social relationships with high collaboration efficiency.

B. Agents

Each agent $a_i \in A$ is defined by a 3-tuple $<Q(a_i), C(a_i), N(a_i)>$, where $Q(a_i) \subseteq S$ indicates the set of skills that agent a_i owns and if $s_j \in Q(a_i)$, agent a_i can provide the skill s_j; $C(a_i) = \{c(a_i, s_1), \ldots, c(a_i, s_{|Q(ai)|})\}$ indicates agent a_i's working cost of providing its skill $s_j \in Q(a_i)$ (where $|X|$ denotes the number of elements in the set X); and $N(a_i)$ indicates the direct social neighbors of a_i, i.e., $N(a_i) = \{a_j | (a_i, a_j) \in E\}$. Furthermore, we assume that each agent cannot join more than one team at a time. However, as a member of a team it can provide more than one skill to team task. This assumption is reasonable since each individual has limited energy and participating in multiple teams will degrade the performance of each team it joins [17], [46].

C. Tasks

We consider a set of tasks $K = \{\kappa_1, \kappa_2, \ldots, \kappa_n\}$ initiated by these agents A independently. Then, each task $\kappa \in K$ can be defined by a tuple $< Ia_\kappa, R_\kappa, \mathrm{It}_\kappa, \mathrm{Dl}_\kappa, \mathrm{Wt}_\kappa, v_\kappa(t) >$, where $Ia_\kappa : \kappa \to A$ indicates the agent that initiates task κ. In this paper, we consider a practical scenario that at each time step, tasks are initiated by agents independently with certain probability. R_κ is the set of skills required by κ and if $s_j \in R_\kappa$, task κ needs the skill service s_j. It_κ is the initialization time of κ, i.e., at time It_κ, κ is generated by Ia_κ. Dl_κ is the deadline (the latest execution start time) of κ, and Wt_κ indicates the working time it will take to execute κ (note that κ must be executed completely before $\mathrm{Dl}_\kappa + \mathrm{Wt}\kappa$). Finally, $v_\kappa(t)$ represents the value associated with the task κ, which is a function of time t. Here, referring to the related definition in [26], we define the value function as

$$v_\kappa(t) = \begin{cases} v_\kappa \left(\dfrac{\mathrm{Dl}_\kappa + \mathrm{Wt}_\kappa - t}{\mathrm{Dl}_\kappa - \mathrm{It}_\kappa} \right)^\delta, & t \le \mathrm{Dl}_\kappa + \mathrm{Wt}_\kappa \\ 0, & t > \mathrm{Dl}_\kappa + \mathrm{Wt}_\kappa \end{cases} \quad (1)$$

where v_κ is the value associated with κ, set by its initiator Ia_κ at the initialization time It_κ; $\delta (0 < \delta \le 1)$ is the parameter modeling how task value decreases with the elapse of time t. If task κ starts execution before its deadline Dl_κ, $v_\kappa(t)$ has the maximum value at time $\mathrm{It}_\kappa + \mathrm{Wt}_\kappa$ and the minimum value at time $\mathrm{Dl}_\kappa + \mathrm{Wt}_\kappa$.

D. Teams

Each team T_κ is responsible for a task κ and is denoted by a 3-tuple $<\Omega_\kappa, \mathrm{Cont}(\Omega_\kappa, \kappa, P_\kappa), us_\kappa>$, where Ω_κ is the set of agents that join T_κ, i.e., the team members of T_κ. $\mathrm{Cont}(\Omega_\kappa, \kappa, P_\kappa) = \{(a_i, s_j, p(a_i, s_j)), \ldots, (a_p, s_q, p(a_p, s_q))\}$ is the skill contribution function indicating that which team member contributes which skill service and how much will be paid. In other words, the tuple $(a_i, s_j, p(a_i, s_j))$ means that team member a_i contributes skill s_j and in return, team manager Ia_κ pays $p(a_i, s_j)$ for a_i's skill provision of s_j. Finally, us_κ is the skills that are not satisfied by the team members Ω_κ, i.e., $us_\kappa = R_\kappa \backslash \{s_j | (a_i, s_j, p(a_i, s_j)) \in \mathrm{Cont}(\Omega_\kappa, \kappa, P_\kappa)\}$. A team T_κ is called a *complete* fulfilled team if each skill has been satisfied by one team member, i.e., $\forall s_j \in R_\kappa, \exists a_i \in \Omega_\kappa : (a_i, s_j, \cdot) \in \mathrm{Cont}(\Omega_\kappa, \kappa, \cdot)$. Otherwise, T_κ is a *partial* fulfilled team.

E. Social Team Formation Problem

Given a task κ and a $\mathrm{SN} = <A, E>$, the team manager Ia_κ first wishes to build a *feasible* team $T_\kappa = <\Omega_\kappa, \mathrm{Cont}(\Omega_\kappa, \kappa, \mathrm{P}_\kappa), \emptyset>$. A feasible team T_κ must satisfy the following three constraints.

1) The team T_κ must be professional. Each skill of κ must be satisfied by one team member, i.e., $\forall s_j \in R_\kappa, \exists a_i \in \Omega_\kappa : (a_i, s_j, \cdot) \in \mathrm{Cont}(\Omega_\kappa, \kappa, \cdot)$.
2) The team T_κ should not include any redundant workers. Each team member must provide at least one skill service, i.e., $\forall a_i \in \Omega_\kappa$ and $s_j \in R_\kappa, \exists (a_i, s_j, \cdot) : (a_i, s_j, \cdot) \in \mathrm{Cont}(\Omega_\kappa, \kappa, \cdot)$.
3) The team T_κ must be collaborative. The subgraph induced by the team members Ω_κ must be connected.

Algorithm 1 Social Team Formation Model (κ, Ia_κ)

1. Initialize $\Omega_\kappa = Ia_\kappa, R_\kappa = R_\kappa \backslash Q(Ia_\kappa), Cont < \Omega_\kappa, \kappa, \cdot >= \emptyset$.
2. Initialize $T_\kappa = <\Omega_\kappa, Cont < \Omega_\kappa, \kappa, \cdot >, R_\kappa >$.
3. **While** $It_\kappa \leq t \leq Dl_\kappa$ /* *t is the team formation time* */
4. **For** each contractor $a_j \in \Omega_\kappa$
5. **For** $a_i \in N(a_j)$
6. Negotiate(Ia_κ,a_j,a_i) and update team T_κ.
7. **If** $\forall s_y \in R_\kappa, \exists a_x \in \Omega_\kappa : (a_x, s, \cdot) \in Cont(\Omega_\kappa, \kappa, \cdot)$
8. Terminate team formation.
9. **End for**
10. **End for**
11. **End while**

Besides satisfying the feasibility property, the requester's ultimate objective is to form the optimal feasible team with the cheapest workers as soon as possible, i.e., maximize $v_\kappa(t) - \sum_{(ai,sj,p(ai,sj))\in \mathrm{Cont}(\Omega\kappa,\kappa,P\kappa)} p(a_i, s_j)$. This is because the task profit is discounted over time, and the earlier the team formed, the earlier the task can be completed, thereby the more profit will be produced for the requester.

IV. Sketch of the Social Team Formation

Before describing the social team formation model, we first define three roles of agents used throughout the team formation process, i.e., *Manager*, *Contractor*, and *Freelancer*.

Definition 1 (Manager, Contractor, and Freelancer): Given a team T_κ for task κ, the agent who initiates κ is called the manager, the agents who join the team T_κ are called the contractor of team T_κ, and other agents are called the freelancer that the team manager Ia_κ can negotiate with.

The social team formation model employed by the team manager Ia_κ to find a set of beneficial freelancers is presented in Algorithm 1.

In step 1, as the initiator of task κ, agent Ia_κ is responsible for contributing all of its available skills to κ and is responsible for building a team $T_\kappa = <Ia_\kappa, \emptyset, R_\kappa \backslash Q(Ia_\kappa)>$ to manage team contractors and recruit new freelancers (step 2). Before the deadline of task κ (step 3), each team contractor $a_j \in \Omega_\kappa$ is responsible for introducing its neighbor freelancers $N(a_j)$ to the team manager Ia_κ. After team contractor a_j's introduction, in step 6, the team manager Ia_κ is able to negotiate with the freelancer $a_i \in N(a_j)$. The negotiation protocol adopted to make a contract between the team manager Ia_κ and the freelancer a_i about which skills to contribute and how much to be paid for a_i's skill provision will be discussed in Section V. Finally, if the team manager Ia_κ has recruited enough contractors such that they can satisfy all of the skill requirements of κ, Ia_κ will terminate the team formation process and start task execution (steps 7 and 8). It should be noted that this social team formation model can always form a connected subgraph of SNs, which satisfies collaboration objective of social team formation.

V. Negotiation Mechanism

The negotiation mechanism, adopted in the team formation model, extends the traditional contract net (CN) protocol [31] by allowing a team manager Ia_κ and a freelancer a_i to reach an agreement on skill provision and skill provision payment. This mechanism mainly consists of three phases.

1) *Offer:* Ia_κ sends an offer to a_i on skill provision and skill provision payment.
2) *Respond:* a_i responds to Ia_κ's offer, such as accept, reject this offer or propose payment improvement.
3) *Confirm:* Ia_κ confirms a final contract with a_i on skill provision and skill payment.

Because of system dynamics, uncertainty, team competition and time factor, it is almost impossible to compute the best strategies of the requesters and workers. Therefore, inspired by the evidence that these factors have an impact on the performance of real-world e-commence applications [25], [26], [28], the proposed negotiation mechanism connects these influential factors indirectly and develops a set of heuristics to approximate agents' optimal strategies. The main contribution of this paper is the strategies designed for the manager to build beneficial feasible teams and for the worker to join the profitable teams, rather than only implementing the contract-net framework. In the following, we will describe these three phases in detail in Sections V-A (i.e., *offer*), V-B (i.e., *Respond*), and V-C (i.e., *Confirm*), respectively.

A. Team Manager Makes Offer to the Freelancer

Before presenting the offer strategy, it is necessary to discuss the definition of the estimated profit that team manager aims to maximize. Due to the dynamics and uncertainty during social team formation, there are many factors correlated with the manager's estimated profit.

1) The estimated value Ev, attached to task κ at time τ, which is given by the average value of finishing the task between the earliest completion time ($\tau + \mathrm{Wt}_\kappa$) and the latest completion time ($\mathrm{Dl}_\kappa + \mathrm{Wt}_\kappa$), that is

$$\mathrm{Ev}(\kappa, \tau) = \frac{\int_{\tau+\mathrm{Wt}_\kappa}^{\mathrm{Dl}_\kappa+\mathrm{Wt}_\kappa} v_\kappa(t)dt}{\mathrm{Dl}_\kappa - \tau}. \tag{2}$$

2) The success rate, Sr of forming a complete team, which is given by the fraction of task κ's skills that have been satisfied by team contracts, i.e., $\mathrm{Sr}(T_\kappa) = 1 - |us_\kappa|/|R_\kappa|$. The larger fraction of skill requirement has been satisfied, the higher possibility a complete team will be formed.
3) Team contractors' remuneration, Re for performing task κ, which is given by

$$\mathrm{Re}(T_\kappa) = \sum_{(a_i,s_j,p(a_i,s_j))\in \mathrm{Cont}(\Omega_\kappa,\kappa,P_\kappa)} p(a_i, s_j) \tag{3}$$

where $p(a_i, s_j)$ is the payment that a_i requires for providing skill service s_j. In practical applications, the worker's working costs for providing skills are often public and from a worker's perspective, letting the requesters know its true information may help him get a beneficial occupation [25]. Therefore, in the *offer* phase, the skill payment $p(a_i, s_j)$ is equal to the public skill working cost $c(a_i, s_j)$.

Algorithm 2 Offer to Fresh Freelancer (Ia_κ, T_κ, a_i, ns, τ)

*/*ns: the set of skills that freelancer a_i should provide;*
*τ: the current time. */*
1. Initialize $th = Ev(\kappa, \tau)/|R_\kappa|, ns = \emptyset$.
2. **For** $s_j \in as(a_i, R_\kappa)$
3. **If** $c(a_i, s_j) \leq th$, **then** $ns = ns \cup s_j$ and $j = j + 1$.
4. **End For**
5. **If** $ns \neq \emptyset$, send the *Offer* $O = <Ia_\kappa, T_\kappa, a_i, ns>$ to a_i.

As the manager of task κ, Ia_κ prefers to employ a freelancer a_i if a_i's participation increases the success rate of forming a complete team and a_i asks for low skill service provision payment. Then, the estimated profit of team manager can be defined as follows.

Definition 2 (Estimated Profit of Team Manager): For a partial fulfilled team $T_\kappa = <\Omega_\kappa, \text{Cont}(\Omega_\kappa, \kappa, P_\kappa), us_\kappa>$ at time step τ, the team manager Ia_κ's estimated profit is

$$Ep(T_\kappa, \tau) = \text{Sr}(T_\kappa) \cdot \text{Ev}(\kappa, \tau) - \text{Re}(T_\kappa). \tag{4}$$

The meanings of the terms Sr, Ev, and Re are defined above.

Given the freelancer a_i that team manager Ia_κ is negotiating with, Ia_κ should consider all of a_i's available skills $as(a_i, R_\kappa) = Q(a_i) \cap R_\kappa$ to compute the optimal skill provision. Different freelancers might have different kinds of available skills, with whom the team manager should negotiate with different strategies. Next, we first divide the freelancers into two different categories (i.e., *fresh freelancer* and *redundant freelancer*) according to his available skills and then present different *offer* strategies for the requester to negotiate with the two kinds of freelancers in Sections V-A1 and V-A2, respectively.

Definition 3 (Fresh Freelancer and Redundant Freelancer): Given a freelancer a_i that the team manager Ia_κ is negotiating with, if a_i's available skills $as(a_i, R_\kappa)$ are exactly what team T_κ lacks, i.e., $as(a_i, R_\kappa) \subseteq us_\kappa$, then a_i is called a fresh freelancer. These available skills of the fresh freelancer are called the fresh skills. Otherwise, if there exists certain available skill $s_x \in as(a_i, R_\kappa)$ that has been satisfied by certain team contractors $a_y \in \Omega_\kappa$, i.e., $as(a_i, R_\kappa) \not\subset us_\kappa$, then a_i is called a redundant freelancer. These available skills that have been satisfied by certain team contractors are called the redundant skills.

A1. Team Manager Makes Offer to Fresh Freelancer

Suppose the team manager Ia_κ is negotiating with a fresh freelancer a_i whose available skills are just what the team T_κ lacks, Ia_κ has to decide which skills to recruit such that these skills' contribution can maximize Ia_κ's estimated profit. We propose a polynomial algorithm for Ia_κ to select the optimal skills of a_i, shown in Algorithm 2.

In step 1 of Algorithm 2, before negotiating with a_i, Ia_κ first initializes some useful variables: the threshold value $\text{th} = \text{Ev}(\kappa, \tau)/|R_\kappa|$ represents how much will the estimated profit be improved by hiring a fresh skill $s_j \in us_\kappa$. This value is used to evaluate whether a skill is worth hiring or not. The set ns stores the skills that T_κ would hire from a_i. In step 2, Ia_κ evaluates a_i's available skills $as(a_i, R_\kappa)$ one by one, and if the cost of skill s_j is not larger than the threshold value th, the manager adds s_j to the skill set ns and proceeds to evaluate the next skill (step 3). Finally, if Ia_κ finds that it is beneficial to recruit a_i by using its skills ns (i.e., $ns \neq \emptyset$), Ia_κ sends an offer $O = <Ia_\kappa, T_\kappa, a_i, ns>$ to a_i for skill acquirement (step 5).

Besides its low $O(l)$ time complexity (l is the number of available skills in the system), Algorithm 2 also guarantees that the selected skills ns maximizes Ia_κ's estimated profit.

Theorem 1: Given a fresh freelancer a_i that the team manager Ia_κ is negotiating with, Algorithm 2 returns the optimal skill provision of a_i that maximizes Ia_κ's estimated profit.

Proof: Before the negotiation, assume that the team T_κ's current team profile $T_\kappa = <\Omega_\kappa, \text{Cont}(\Omega_\kappa, \kappa, P_\kappa), us_\kappa>$ and the team manager Ia_κ's current estimated profit is

$$\begin{aligned} Ep(T_\kappa, \tau) &= \text{Sr}(T_\kappa) \cdot \text{Ev}(\kappa, \tau) - \text{Re}(T_\kappa) \\ &= \text{Ev}(\kappa, \tau) - |us_\kappa| \cdot \text{th} \\ &\quad - \sum_{(a_i, s_j, p(a_i, s_j)) \in \text{Cont}(\Omega_\kappa, \kappa, P_\kappa)} p(a_i, s_j) \end{aligned} \tag{5}$$

where the threshold value th represents the estimated profit improvement per skill, i.e., $\text{th} = \text{Ev}(\kappa, \tau)/|R_\kappa|$. Then we can derive that the marginal benefit from adding a fresh skill is exactly the threshold and adding a skill that has larger cost than th will definitely decrease Ia_κ's estimated profit. Now we can conclude that for a given fresh freelancer a_i, Algorithm 2 always returns a_i's optimal skill provision. ■

A2. Team Manager Makes Offer to Redundant Freelancer: For a redundant freelancer a_i that agrees to contribute a redundant skill s_j that has been satisfied by the certain team contractor $a_x \in \Omega_\kappa$, Ia_κ needs to remove a_x's skill contribution of s_j. If removing a_x's contribution of skill s_j makes a_x do not contribute skills any more, Ia_κ then will break the contract with a_x. Moreover, if a_y is only connected with the team contractor a_x and removing a_x from team T_κ makes other team contractor $a_y \in \Omega_\kappa$ be isolate, a_y will also depart from T_κ, which further degrades the estimated profit of team T_κ.

Thus, given a redundant freelancer a_i, on one hand, even if a_i's available skill $s_j \in as(a_i, R_\kappa)$ has been satisfied by certain team contractor a_x, s_j still needs to be accounted for. This is because the freelancer a_i might require a lower payment for providing s_j. On the other hand, even if the available skill s_j requires a lower payment than that of the existing team contractor (e.g., a_x) for providing s_j, team manager Ia_κ should not replace the existing skill contribution $(a_x, s_j, \cdot)$ by the cheaper skill contribution $(a_i, s_j, \cdot)$. This is because this kind of replacement might make existing team contractors depart. At first glance, Ia_κ needs to consider all of the exponential $O(2^{|as(ai, Rk)|})$ possible skill combinations to identify the optimal skill contribution of a_i. In the following, we propose an efficient polynomial algorithm (i.e., Algorithm 3) to find the optimal skill contribution of a_i that produces the maximal estimated benefit for Ia_κ.

In Algorithm 3, step 1, before negotiating with the redundant freelancer a_i, the manager Ia_κ first initializes some useful variables, such as max, denoting the current team T_κ's estimated profit and the threshold th, denoting how much will be improved on current team's estimated value by contributing

Algorithm 3 Offer to Redundant Freelancer (Ia_κ, T_κ, a_i, ns, τ)

1. Initialize $q = 1$, $H = \emptyset$, $max = Ep(T_\kappa, \tau)$, $th = Ev(\kappa, \tau)/|R_\kappa|$.
2. Rank $s_j \in as(a_i, R_\kappa)$ such that $c(a_i, s_1) \leq \ldots \leq c(a_i, s_{|as(ai,R\kappa)|})$.
3. **While** $q \leq |as(a_i, R_\kappa)|$
4. **If** $Ep(T_\kappa \otimes \{ns \cup s_q\}, \tau) \geq max$
5. $ns = ns \cup s_q$, $max = Ep(T_\kappa \otimes ns, \tau)$.
6. **Else**
7. **If** $s_q \notin us_\kappa \&\& c(a_i, s_q) \leq min\{th, p(a_x, s_q)\}$
   /*a_x is the team contractor that provides skill s_q*/
8. Set $rs = T_\kappa \backslash \{T_\kappa \otimes \{ns \cup s_q\}\}$ and $cs = \cup_{q<p\leq|as(ai,R\kappa)|} s_p \cap rs$.
9. **For** $s_p \in cs$
10. **If** $c(a_i, s_p) \leq th$, **then** $H = H \cup s_p$;
11. **End for**
12. **If** $Ep(T_\kappa \otimes \{ns \cup s_q \cup H, \tau) \geq max$, **then** $ns = ns \cup s_q \cup H$;
13. **End Else**
14. $q = q + 1$;
15. **End while**
16. If $ns \neq \emptyset$, send the *Offer* $O = <Ia_\kappa, T_\kappa, a_i, ns>$ to a_i.

Algorithm 4 Function ($T_\kappa \otimes cs$)

/*cs: the set of skills that the team T_κ hires and this function returns the team configuration after team T_κ hires skill set cs*/

1. Manager Ia_κ adds a_i's each contributed skill $s_y \in cs$ to the current skill contribution function $Cont(\Omega_\kappa, \kappa, P_\kappa)$.
2. Remove the team contractor a_x's overlapped skill contribution (a_x, s_y), where $s_y \in cs$.
3. Remove the team contractor a_x that does not contribute any skill after removing its skill contribution of s_y.
4. Remove the team contractors (as well as their contributed skills) that are isolate in the team T_κ after removing certain team contractors removed in step 3.

each fresh skill. This variable th is used to evaluate whether a_i's skill is profitable. In step 2, Ia_κ first sorts a_i's available skills $as(a_i, R_\kappa)$ in increasing order of working cost and then evaluates these skills in the order of their rank (steps 3–15). In step 4, we use $Ep(T_\kappa \otimes \{ns \cup s_q\}, \tau)$ to represent Ia_κ's estimated profit after the redundant freelancer a_i contributes skills $\{ns \cup s_q\}$. In this case that a_i is a redundant freelancer, $Ep(T_\kappa \otimes \{ns \cup s_q\}, \tau)$ can be computed by Algorithm 4, where Ia_κ first adds a_i's each contributed skill $s_y \in \{ns \cup s_q\}$ to the current skill contribution function $\text{Cont}(\Omega_\kappa, \kappa, \cdot)$ (step 1 of Algorithm 4). And then Ia_κ updates team configuration $T_\kappa \otimes \{ns \cup s_q\}$ by removing existing team contractors' overlapping skill contribution (step 2 of Algorithm 4), removing the team contractors that do not contribute any skill after removing their overlapping skill contribution (step 3 of Algorithm 4), and finally removing the team contractors that are isolate from the team T_κ (step 4 of Algorithm 4). If the estimated profit produced by the updated team $T_\kappa \otimes \{ns \cup s_q\}$ in not less than that of the previous team $T_\kappa \otimes ns$ with the estimated profit max (step 4), add s_q to ns (step 5). Otherwise, if a_i's working cost for s_q, $c(a_i, s_q)$ is not greater than the minimum between the value th and the skill payment $p(a_x, s_q)$ of the team contractor a_x (step 7), then the only reason for the updated team $T_\kappa \otimes \{ns \cup s_q\}$ produces a lower estimated profit than that of the previous team $T_\kappa \otimes ns$ is that a_i's skill contribution of s_q will make the contractor a_x and other contractors (if any) depart. These team contractors' departure will make their contributed skills removed in T_κ, thereby decreasing the estimated profit of team $T_\kappa \otimes \{ns \cup s_q\}$. Therefore, in the following steps (steps 8–11), we try to compensate for these removed skills by using the remaining available skills $\cup_{q<p\leq|as(ai,R\kappa)|} s_p$ of a_i. First, in step 8, denoted by $rs = T_\kappa \backslash \{T_\kappa \otimes \{ns \cup s_q\}\}$ as the removed skills because of a_i's skill contribution of s_q and by $cs = \cup_{q<p\leq|as(ai,R\kappa)|} s_p \cap rs$ as the remaining available skills of a_i that can be used to compensate for the removed skills rs. Here, we can derive that cs is exactly what the current team $T_\kappa \otimes \{ns \cup s_q\}$ lacks. Then be similar to the scenario of negotiating with a fresh freelancer, we can utilize the idea of Algorithm 2 to identify the optimal compensation skills (i.e., H) from the available compensation skill set cs (steps 9–11). If the value of $Ep(T_\kappa \otimes \{ns \cup s_q \cup H, \tau)$ by contributing the compensation skills H to team $T_\kappa \otimes \{ns \cup s_q\}$ is not less than max, Ia_κ will consider hiring a_i's skill s_q together with the compensation skills H (step 12). Finally, if Ia_κ finds it is beneficial to hire a_i by using its skills ns (i.e., $ns \neq \emptyset$), Ia_κ then sends an offer to a_i for skill acquirement (step 16).

Before presenting the optimization of Algorithm 3 (which is given in Theorem 2), we first give the following two lemmas that are useful to prove Theorem 2.

Lemma 1: Given a team T_κ and a redundant freelancer a_i with the available skills $Z = X \cup Y$ ($X, Y \neq \emptyset$ and $X \cap Y = \emptyset$), then $Ep(T_\kappa \otimes Z) = Ep(T_\kappa \otimes X) + Ep(T_\kappa \otimes Y) - Ep(T_\kappa)$, where $T_\kappa \otimes X$ is the updated team of T_κ by contributing the skills X to T_κ (defined in Algorithm 4).

Proof: Based on the definitions of team manager's estimated profit (i.e., Ep) and team updating function $T_\kappa \otimes Z$ (i.e., Algorithm 4), we can derive the estimated profit by contributing the skill set X

$$Ep(T_\kappa \otimes X, \tau) = Ep(T_\kappa, \tau) + (|X| - |rs_X|) \cdot \text{th} - \left(\sum_{s_j \in X} c(a_i, s_j) - \sum_{s_j \in rs_X} p(\cdot, s_j) \right) \quad (6)$$

where $\text{th} = \text{Ev}(\kappa, \tau)/|R_\kappa|$ is the estimated profit improvement per skill contribution, and rs_x indicates the skills removed from team T_κ because of the contribution of skills X's. Since $X \cap Y = \emptyset$, then we have

$$\begin{aligned} &Ep(T_\kappa \otimes X, \tau) + Ep(T_\kappa \otimes Y, \tau) \\ &\quad = 2Ep(T_\kappa, \tau) + (|X| + |Y| - |rs_X| - |rs_Y|) \cdot \text{th} \\ &\qquad - \left(\sum_{s_j \in X \bigcup Y} c(a_i, s_j) - \sum_{s_j \in rs_Y \bigcup rs_X} p(\cdot, s_j) \right) \\ &\quad = Ep(T_\kappa, \tau) + Ep(T_\kappa \otimes Z, \tau). \end{aligned} \quad (7)$$

Therefore, we can conclude $Ep(T_\kappa \otimes Z) = Ep(T_\kappa \otimes X) + Ep(T_\kappa \otimes Y) - Ep(T_\kappa)$. Here for simplicity, we omit the symbol τ. ■

Lemma 2: Given a team T_k and a redundant freelancer a_i with the available skills $Z = \cup_{1\leq i\leq n} X_i (\forall X_j, X_k : X_j, X_k \neq \emptyset$ and $X_j \cap X_k = \emptyset)$, then $Ep(T_\kappa \otimes Z) = \sum_{1\leq i\leq n} Ep(T_\kappa \otimes X_i) - (n-1)Ep(T_\kappa)$, where $T_\kappa \otimes X_i$ is the updated team of T_κ by contributing the skill set X_i to T_κ.

Proof: According to Lemma 1, we can derive

$$\begin{cases} Ep(T_\kappa \otimes Z) = Ep(T_\kappa \otimes \bigcup_{1\le i\le n-1} X_i) \\ \qquad + Ep(T_\kappa \otimes X_n) - Ep(T_\kappa) \\ Ep(T_\kappa \otimes \bigcup_{1\le i\le n-1} X_i) = Ep(T_\kappa \otimes \bigcup_{1\le i\le n-2} X_i) \\ \qquad + Ep(T_\kappa \otimes X_{n-1}) - Ep(T_\kappa) \\ \vdots \\ Ep(T_\kappa \otimes \{X_1 \bigcup X_2\}) = Ep(T_\kappa \otimes X_1) \\ \qquad + Ep(T_\kappa \otimes X_2) - Ep(T_\kappa). \end{cases} \tag{8}$$

Summing all of the above $n-1$ equations in (8), we have $Ep(T_\kappa \otimes Z) = \sum_{1\le i\le n} Ep(T_\kappa \otimes X_i) - (n-1)Ep(T_\kappa)$. ■

Now we are ready to give the optimization of Algorithm 3.

Theorem 2: Given a redundant freelancer a_i that Ia_κ is negotiating with, Algorithm 3 returns the optimal skill contribution of a_i that produces the maximal estimated profit for Ia_κ.

Proof: Suppose that the skill set returned by Algorithm 3 is Alg, and the skill set returned by the optimum is Opt. In the following, we will prove that Opt = Alg. Denoted by Θ as the common contributed skills between Alg and Opt, i.e., $\Theta = \text{Alg} \cap \text{Opt}$, by θ as the skills contributed by Alg only, i.e., $\theta = \text{Alg}\backslash\Theta$ and by ψ as the skills contributed by Opt only, i.e., $\psi = \text{Opt}\backslash\Theta$. To prove Opt = Alg, we only need to prove $\theta = \psi = \emptyset$.

Conclusion 1 ($\psi = \emptyset$): We achieve this conclusion by proving that if the skills contributed only by Opt (i.e., ψ) increase the estimated profit of the team of $T_\kappa \otimes \Theta$, the team manager Ia_κ can always find the skills ψ that increases the estimated profit of the team $T_\kappa \otimes \text{Alg}$. As described in Algorithm 3, the skills ψ are first sorted in increasing order of their working cost such that $c(a_i, s_1) \le \cdots \le c(a_i, s_{|\psi|})$ and then are evaluated one by one in their order. For the first skill $s_1 \in \psi$, if Algorithm 3 does not select s_1, we have $Ep(T_\kappa \otimes \{\text{Alg} \cup s_1\}) < Ep(T_\kappa \otimes \text{Alg})$ (Assumption 1), otherwise, Algorithm 3 will identify this beneficial skill. According to Algorithm 3, we can derive that there are three possible cases to support Assumption 1.

Case 1 [$s_1 \in us_\kappa \&\& c(a_i, s_1) > th$]: This is impossible, because it is profitable for Opt to select the unsatisfied skill s_1, where $c(a_i, s_1) \le \text{th}$.

Case 2 [$s_1 \notin us_\kappa \&\& c(a_i, s_1) > \min\{th, c(a_x, s_1)\}$ (a_x is the Team Contractor That Has Agreed to Contribute s_1)]: This is also impossible, because contributing s_1 will also decreases the estimated profit of team $T_\kappa \otimes \{\text{Opt}\backslash s_1\}$, which contradicts the fact that Opt is the optimal skill contribution.

Case 3 ($s_1 \notin us_\kappa \&\& c(a_i, s_1) < min\{th, c(a_x, s_1)\}$): In this case, the reason that contributing s_1 does not improve the team profit $Ep(T_\kappa \otimes \text{Alg})$ is that the skill contribution of s_1 makes other satisfied skills $rs \subseteq \text{Alg}\backslash s_1$ removed, thereby decreasing the estimated profit of team $T_\kappa \otimes \text{Alg}$.

Next, we will prove that case 3 never happens. If the skill contribution of s_1 to team $T_\kappa \otimes \text{Alg}$ makes other satisfied skills *rs* removed, then by steps 9–12, Algorithm 3 can always find the optimal compensation skills $cs \subseteq \{\psi\backslash s_1\}$ for team $T_\kappa \otimes \{\text{Alg} \cup s_1\}$ (which has been proved in Theorem 1). By Assumption 1, this contributions of skill s_1 associated with the compensation skills *cs* do not increase the estimated profit of team $T_\kappa \otimes \text{Alg}$, that is

$$\begin{aligned} &Ep(T_\kappa \otimes \{\text{Alg} \cup s_1 \cup cs\}) < Ep(T_\kappa \otimes \text{Alg}) \\ &\quad \Rightarrow Ep(T_\kappa \otimes \text{Alg}) + Ep(T_\kappa \otimes \{s_1 \cup cs\}) - Ep(T_\kappa) \\ &\qquad < Ep(T_\kappa \otimes \text{Alg}) \\ &\quad \Rightarrow Ep(T_\kappa \otimes \{s_1 \cup cs\}) - Ep(T_\kappa) < 0. \end{aligned} \tag{9}$$

The inequality (9) is derived from Lemma 2. On the other hand, from the view of Opt, we have

$$\begin{aligned} Ep(T_\kappa \otimes \text{Opt}) &= Ep(T_\kappa \otimes \{\Theta \cup \{s_1 \cup cs\} \cup \{\psi\backslash\{s_1 \cup cs\}\}) \\ &= Ep(T_\kappa \otimes \Theta) + Ep(T_\kappa \otimes \{\psi\backslash\{s_1 \cup cs\}\}) \\ &\quad + Ep(T_\kappa \otimes \{s_1 \cup cs\}) - 2Ep(T_\kappa) \\ &< Ep(T_\kappa \otimes \Theta) + Ep(T_\kappa \otimes \{\psi\backslash\{s_1 \cup cs\}\}) \\ &\quad - Ep(T_\kappa). \end{aligned} \tag{10}$$

However, as Opt is the optimal skill provision, we have

$$\begin{aligned} Ep(T_\kappa \otimes \text{Opt}) \ge\ & Ep(T_\kappa \otimes \{\Theta \cup \{\psi\backslash\{s_1 \cup cs\}\}\}) \\ \Rightarrow\ & Ep(T_\kappa \otimes \text{Opt}) \ge Ep(T_\kappa \otimes \Theta) \\ & + Ep(T_\kappa \otimes \{\psi\backslash\{s_1 \cup cs\}\}) - Ep(T_\kappa). \end{aligned} \tag{11}$$

The inequality (11) holds because $\Theta \cup \{\psi\backslash\{s_1 \cup cs\}\} \subseteq \text{Opt}$ and $\Theta \cap \{\psi\backslash\{s_1 \cup cs\}\} = \emptyset$, which contradicts inequality (10). Therefore, we can conclude that $\psi = \emptyset$.

Conclusion 2 ($\theta = \emptyset$): The proof of this conclusion is similar to that for conclusion 1 and due to the limitations of space, we omit the proof. ■

B. Freelancer Makes Response to the Team Manager

Once the freelancer a_i receives the offer $O = <Ia_\kappa, T_\kappa, a_i, ns>$ from team manager Ia_κ, a_i assesses this offer and make a response to Ia_k. Before describing a_i's response strategy, we first define the states (i.e., *Fully-contracted*, *Partial-contracted*, and *Free*) of a_i during social team formation.

Definition 4 (States of Agents): During social team formation, the agent who initiates a task or has been a member of a fulfilled team is in state Fully-contracted; the agent who has been a member of a partial team is in state Partial-contracted; and the agent who neither initiates a task nor has joined a team is in state Free.

Agents in different states might have different response strategies. In the following, we will present a_i's optimal response strategies within different states.

Case 1 (a_i is Free): In this case, a_i will accept team T_κ's offer. This is because joining a team to work on team task can obtain some financial remuneration, which is a rather economical option compared to state in free where there is no payment. Moreover, in the Free state, the payment $p(a_i, s_j)$ required for providing a_i's skill service s_j is just his public working cost $c(a_i, s_j)$, i.e., $p(a_i, s_j) = c(a_i, s_j)$.

Case 2 (a_i is Partial-Contracted): Assume that a_i has been a member of a partial team $T_{\kappa*}$ and now at time step τ, a_i receives a new offer $O = <Ia_\kappa, T_\kappa, a_i, ns>$ from a new team T_κ. A dilemma is faced by a_i: breaking the contract with the original team $T_{\kappa*}$ by joining this new team T_κ or staying with the original team $T_{\kappa*}$ by rejecting the new offer $O(\cdot, T_\kappa, \cdot)$.

Here, to quantify how much a_i gains by accepting or rejecting, the measure of estimated remuneration (per unit time) is utilized. On one hand, staying in the original team $T_{\kappa *}$, a_i will obtain the estimated remuneration

$$Er\big(a_i, T_{\kappa_*}, \tau\big) = \frac{\mathrm{Sr}\big(T_{\kappa_*}, \tau\big) \sum_{(a_i, s_j, p(a_i, s_j)) \in \mathrm{Cont}(\Omega_{\kappa *}, \kappa_*, P_{\kappa *})} p\big(a_i, s_j\big)}{\mathrm{Wt}_{\kappa_*}}. \quad (12)$$

The formula (12) indicates that a_i's estimated remuneration (per unit time) $Er(a_i, T_{\kappa *}, \tau)$ is as follows.

1) Directly proportional to the success rate, $\mathrm{Sr}(T_{\kappa *}, \tau)$ of forming a complete team $T_{\kappa *}$, which is further positively related to the skills that have been satisfied and the remaining time for building team, i.e., $\mathrm{Sr}(\mathrm{T}_{\kappa *}, \tau) = (1 - |us_{\kappa *}|/|R_{\kappa *}|)(\mathrm{Dl}_{\kappa *} - \tau)$.
2) Directly proportional to the revenue achieved by providing his skill service, i.e., $\sum_{(a_i, s_j, p(a_i, s_j)) \in \mathrm{Cont}(\Omega_{k *}, \kappa_*, P_{k *})} p(a_i, s_j)$.
3) Inversely proportional to the working time, $\mathrm{Wt}_{\kappa *}$, that is required to accomplish the task κ_*.

On the other hand, joining the new team T_κ that sends the new offer $O = <Ia_\kappa, T_\kappa, a_i, ns>$, a_i will obtain the estimated remuneration (per unit time)

$$Er(a_i, T_\kappa, ns, \tau) = \frac{\mathrm{Sr}(T_\kappa, \tau) \sum_{s_j \in ns} c\big(a_i, s_j\big)}{\mathrm{Wt}_\kappa} \quad (13)$$

where $\mathrm{Sr}(T_\kappa, \tau) = (1 - |us_\kappa \backslash ns|/|R_\kappa|)(\mathrm{Dl}_\kappa - \tau)$ indicates the success rate of the new team T_κ that will be formed completely if a_i agrees to contribute the skills ns. The value $\sum_{sj \in ns} c(a_i, s_j)$ indicates the revenue a_i will achieve by joining T_κ. The meanings of other terms are similar to those discussed in (12).

As a rational freelancer that aims to maximize his own payment, a_i prefers to join the team that can achieve a high estimated remuneration. Now we are ready to discuss the corresponding response strategy of a_i by comparing the values of $Er(a_i, T_{\kappa *}, \tau)$ and $Er(a_i, T_\kappa, ns, \tau)$.

1) *If $Er(a_i, T_{\kappa *}, \tau) \geq Er(a_i, T_\kappa, ns, \tau)$:* In the case that a_i's estimated remuneration achieved from the original team $T_{\kappa *}$ is not less than that of the new team T_κ, a_i prefers to stay in the original team $T_{\kappa *}$ and reject the offer of T_κ.
2) *If $Er(a_i, T_{\kappa *}, \tau) < Er(a_i, T_\kappa, ns, \tau)$:* In this case, intuitively, a_i should break the contract with the original team $T_{\kappa *}$ and join the new team T_κ. However, the team contractor a_i's departure will make the original team $T_{\kappa *}$ be disconnected. This disconnection will make other team contractors isolate and depart, which will degrade the estimated profit of $T_{\kappa *}$. Therefore, when considering to terminate the contract with his joined team $T_{\kappa *}$, a_i should attempt to propose whether the team manager $Ia_{\kappa *}$ is willing to improve the payments of the contributed skills to persuade him to stay with the team. Here, denoted by cs as a_i's contributed skills to his joined team $T_{\kappa *}$, i.e., $cs(a_i, \mathrm{T}_{\kappa *}) = \{s_j | (a_i, s_j, \cdot) \in \mathrm{Cont}(\Omega_{\kappa *}, \kappa_*, \cdot)\}$. Then, to achieve as high estimated remuneration as that would be achieved from the new team T_κ, a_i will propose to improve the payment of each contributed skill $s_j \in cs$ to

$$p'\big(a_i, s_j\big) = Er(a_i, T_\kappa, ns, \tau) \times \frac{\mathrm{Wt}_{\kappa_*}}{\mathrm{Sr}(T_{\kappa_*}, \tau)} \times \frac{p(a_i, s_j)}{\sum_{s_x \in cS} p(a_i, s_x)} \quad (14)$$

where $Er(a_i, T_\kappa, ns, \tau)$ is the estimated remuneration that would be achieved from the new team T_κ computed by (13). The values $p(a_i, s_j)$ and $p'(a_i, s_j)$ are the original and adjusted skill payment, respectively. After computing the adjusted payment, a_i will send a payment improvement response $R = <\{p(a_i, s_j) \rightarrow p'(a_i, s_j) | s_j \in cs(a_i, T_{\kappa *})\}>$ to the original team's manager $Ia_{\kappa *}$.

Case 3 (a_i is Fully-Contracted): If a_i's joined team T_κ has hired enough contractors such that they can satisfy all of the skill requirements of team task κ and has started task execution, a_i will reject any new offer until task κ has been finished. This is because each contractor can only make contract with one team and breach the current complete fulfilled team will suffer tremendous monetary penalty or reputation loss [26], [42].

C. Team Manager Makes Confirmation to the Freelancer

As the manager of T_κ, Ia_κ might receive two kinds of responses: 1) the payment improvement response from his team contractors because of other teams' competition and 2) the acceptance or rejection response from the freelancer. Next we will discuss the corresponding confirmation strategy of Ia_κ when he confronts these two kinds of responses.

Case 1: For the contractor a_i that proposes to improve skill payment $<\{p(a_i, s_j) \rightarrow p'(a_i, s_j) | s_j \in cs(a_i, T_{\kappa *})\}>$ (cs represents the skills that a_i agrees to provide to his joined team T_κ), the team manager Ia_κ will compare the estimated profit of team profile $Ep(T_\kappa, \{p(a_i, s_j) \rightarrow p'(a_i, s_j) | s_j \in cs(a_i, T_{\kappa *})\}, \tau)$ by improving a_i's skill payment and the estimated profit $Ep(T_\kappa \backslash \{a_i\}, \tau)$ of the team profile $T_\kappa \backslash \{a_i\}$ by removing the team contractor a_i. If $Ep(T_\kappa, \{p(a_i, s_j) \rightarrow p'(a_i, s_j) | s_j \in cs(a_i, T_{\kappa *})\}, \tau) \geq Ep(T_\kappa \backslash \{a_i\}, \tau)$, Ia_κ will accept a_i's proposal on skill payment improvement. Otherwise, he will reject a_i's proposal.

Case 2: For the freelancer a_i that sends the acceptance or rejection response, the team manager Ia_κ needs to make some agreements for this response. On one hand, if a_i accepts Ia_κ's offer $O = <Ia_\kappa, T_\kappa, a_i, ns>$, Ia_κ first makes a *tentative* contract with a_i on skill contribution ns and skill payment. A tentative contract means that before T_κ is formed completely, team contractor a_i can adjust its skill contribution to T_κ as well as can propose to improve the skill payment. Furthermore, if a_i's skill contribution leads to a complete fulfilled team for task κ, all tentative contracts of this team will become a *final* contract such that team contractors cannot breach the contract unilaterally until κ is finished successfully. After making the contract with a_i, Ia_κ then updates team configuration by adding the skill contributions $\cup_{sj \in ns}(a_i, s_j, p(a_i, s_j))$ to $\mathrm{Cont}(\Omega_\kappa, \kappa, P_\kappa)$, removing the overlapped skill contributions provided by other team contractors, removing the team contractors that do not contribute skills

Algorithm 5 Negotiate(Ia_κ, a_j, a_i)

Stage 1: Manager Ia_κ calls Algorithm 2 (i.e., a_i is a fresh freelancer) or Algorithm 3 (i.e., a_i is a redundant freelancer) to generate the offer $O = <Ia_\kappa, T_\kappa, a_i, ns>$ to a_i.
Stage 2:
1) **If** a_i is *Free*
2) Freelancer a_i sends response $R = <acceptance>$ to Ia_κ.
3) **If** a_i is *Fully-contracted*
4) Freelancer a_i sends $R = <rejection>$ to Ia_κ.
5) **If** a_i is *Partially-contracted*
6) **If** $Er(a_i, T_{\kappa^*}, \tau) \geq Er(a_i, T_\kappa, ns, \tau)$
/* $T_\kappa{}^*$ *is the current team of which* a_i *is a member* */
7) Freelancer a_i sends $R = <rejection>$ to Ia_κ.
8) **Else**
Freelancer a_i first proposes to his joined team manager $Ia_\kappa{}^*$ for payment improvement $\{s_j \in cs(a_i, T_{\kappa^*}) | p(a_i, s_j) \rightarrow p'(a_i, s_j)\}$ (computed by formula (15)).
9) **If** $Ia_\kappa{}^*$ agrees to improve a_i's payment
10) Freelancer a_i sends $R = <rejection>$ to Ia_κ.
11) **Else** Freelancer a_i sends $R = <acceptance>$ to Ia_κ.
Stage 3: If $R = <acceptance>$, Ia_κ makes a contract (*tentative* or *final*) with a_i and updates team configuration.

anymore and removing those isolate team contractors as well as these contractors' skills. On the other hand, if a_i rejects Ia_κ's offer $O = <Ia_\kappa, T_\kappa, a_i, ns>$, Ia_κ does nothing.

D. Negotiation Algorithm

Up to this point, we have described the negotiation process between a team manager and a freelancer. A formal description of this negotiation mechanism that makes an agreement on skill provision and skill provision payment between team manager Ia_κ and freelancer a_i is shown in Algorithm 5. Algorithm 5 is implemented in a distributed manner such that each requester/worker invokes the negotiation mechanism independently. Referring to [50] for a more detail description on how to construct these networks.

VI. Experimental Validation and Analyses

In Section VI-A, we compare the proposed social team formation model with the traditional distributed models on social welfare and team formation time. In Section VI-B, we compare our model with the benchmark centralized model on social welfare.

A. Comparing With the Traditional Distributed Social Team Formation Models

1) Experimental Setting:

a) Dataset: We collect the data on 764 workers registered in the crowdsourcing website Upwork, upwork.com. For each worker, we record two kinds of personal information: 1) the set of skills he owns and 2) the salary in dollars per hour he requires. Through analyzing these workers' information, we observe that there are 20 kinds of skills available by these workers, the skill number of each worker distributes in the range [1, 13] and the cost of providing each skill service distributes in [0.5, 90] randomly. These collected workers are interconnected by three typical SN structures such as random network (Random), small-world network (SMW) and scale-free network (SF).

TABLE II
Parameter Setup

Parameter	Value	Definitions
n	746	The number of agents in SN
l	20	The number of skills in SN
$\|Q(a_i)\|$	[1, 13]	The number of skills of a_i
$c(a_i,s_j)$	[0.5, 90]	The cost of a_i providing skill s_j
d	[6, 20]	The network degree
λ	[0.1, 0.9]	The task arrival rate
π	1000	The system running time steps
It_κ	[0,π]	The initialization time of task κ
Dl_κ	[It_κ,It_κ+50]	The deadline of task κ
Wt_κ	[20, 40]	The working time of task κ
v_κ	[200, 400]	The value of task κ
δ	[0.2, 1.6]	Task discount rate

b) Parameter setting: By referring to the related definition on parameter setting in [26], we range other parameters involved in the social team formation process as follows. At each time step, a task arrives at the system with a probability $\lambda = 0.1 \sim 0.9$. The period required to accomplish a task (i.e., Wt_κ) is drawn from $U(20, 40)$ [$U(a, b)$ returns the value in the range $[a, b]$ randomly] and the deadline of each task (i.e., Dl_κ) is distributed in $[\mathrm{It}_\kappa, \mathrm{It}_\kappa+50]$, which must be greater than the task's initiation time It_κ. The value associated with each task (i.e., v_κ) is distributed in $U(200, 400)$ and the discount rate on time decay (i.e., δ) is distributed in [0.2, 1.6]. For clarity, we give a detail description of the used variables in Table II.

c) Comparison models and performance metrics: We compare the designed social team formation model (our model) with other two distributed models, i.e., the greedy model (Greedy) and CN model.

1) *Greedy Model [28]:* In this model, the task manager evaluates the skills of the negotiated worker a_i one by one and once he finds a_i has lower working cost of providing s_j than that of certain team member $a_x \in \Omega_\kappa$, i.e., $c(a_i, s_j) < c(a_x, s_j)$, the manager replaces the skill contribution $(a_x, s_j, c(a_x, s_j))$ by using a_i's skill service s_j. On the other hand, the workers only join the team where they can achieve high payment. Compared to this model, our model's advantage of considering the combinational skill contribution could be revealed.
2) *CN Model [31]:* In this model, the task manager only selects the workers that have the skills the team lacks and the workers only join the team where they can achieve high payment. Since the negotiated workers always contribute the team's lacked skills, there is no conflict between the negotiated worker and team contractors, therefore the formed team is always connected. Compared to this model, our model's advantage of negotiating the working cost could be revealed.

We evaluate the performance of these models through social welfare (SW) and the time used for team formation. SW is defined as the sum of all managers' profits, that is

$$\mathrm{SW} = \sum_{i=1}^{n} \mathrm{profit}(Ia_{\kappa_i}) = \sum_{\kappa \in CT} \left(v_\kappa(ct_\kappa) - \mathrm{Re}(T_\kappa)\right) \quad (15)$$

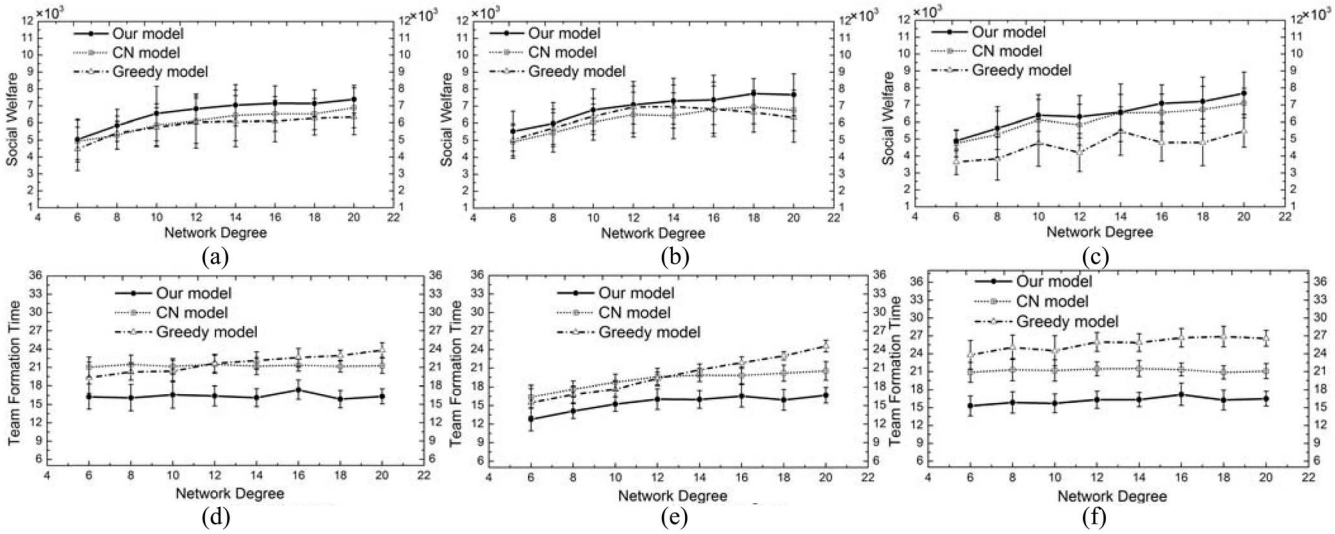

Fig. 1. Effect of network degree on different social team formation models' SW (a–c) and team formation time (d–f). (a) and (d) Random. (b) and (e) SMW. (c) and (f) SF.

TABLE III
PROPERTIES OF NETWORKS

Property / Network	Network Degree											
	=6			=10			=14			=18		
	Dia	*Aspl*	*Clu*	*Dia*	*Aspl*	*Clu*	*Dia*	*Aspl*	*Clu*	*Dia*	*Aspl*	*Clu*
Random	7.0	4.0	0.007	5.1	3.3	0.011	4.9	3.0	0.015	4.0	2.7	0.019
Small-World	7.5	4.5	0.22	5.2	3.5	0.24	5. 0	3.1	0.25	4.0	2.8	0.26
Scale-Free	6.0	3.5	0.03	5.0	2.9	0.035	4.0	2.7	0.045	4.0	2.5	0.053

where CT denotes the set of tasks that are completed successfully and ct_κ is the completion time of task κ and $\mathrm{Re}(T_\kappa)$ indicates the total remuneration paid to team workers.

We perform a series of experiments to validate our model: we first test the effect of network degree and task discount rate on the distributed social team formation models' SW and team formation time in Sections VI-A2a and VI-A2b, respectively. In Section VI-B, we test the SW of our model by comparing our model with the benchmark centralized model. All the results plotted in the figure are recorded by averaging over 50 instances.

2) Experimental Results:

a) Effect of network degree on social welfare and team formation time: Fig. 1 shows the SW [Fig. 1(a)–(c)] and the team formation time [Fig. 1(d)–(f)] achieved by our model, CN and Greedy on network degree, where task discount rate is set to 0.8. The network degree is computed as the average degree of all workers. From Fig. 1, we have the following observations.

1) Within all network structures from Fig. 1(a)–(c), our model produces larger SW than the other two distributed Greedy and CN models. This can be explained from two perspectives.
   a) As shown in Fig. 1(d)–(f), our model uses less time on team formation. Because the task value is discounted over time, the less time used for team formation, the earlier the task will be completed, and then the larger task profit will be produced.
   b) Compared to CN model, the team manager in our model negotiates with the workers on working cost, which can help the manager build team of inexpensive workers. Although the team manager in Greedy also evaluates the worker's working cost, its large team formation time prevents Greedy producing high task profit.
2) The SW produced by Greedy performs much worse in the networks of SF compared to the SWs in Random and SMW. This can be explained from the perspective of team formation time. Compared Fig. 1(d) and (e) with Fig. 1(f), we can observe that the team formation time of Greedy in SF is larger than the team formation time of Random (or SMW). However, why does Greedy generate larger team formation time in SF than that in Random (or SMW)? We can explain this phenomenon by the network property. Table III shows the three typical network properties, such as network diameter (*Dia*), average shortest path length (*Aspl*) and clustering coefficient (*Clu*) ([50] gives a detail illustration of these properties). From Table III, we can find that the SF network have shorter *Dia* and *Aspl* than those of random (or SMW). The shorter *Dia* and *Aspl* indicate that the workers can interact with others more easily. Now we will explain why Greedy generates the larger

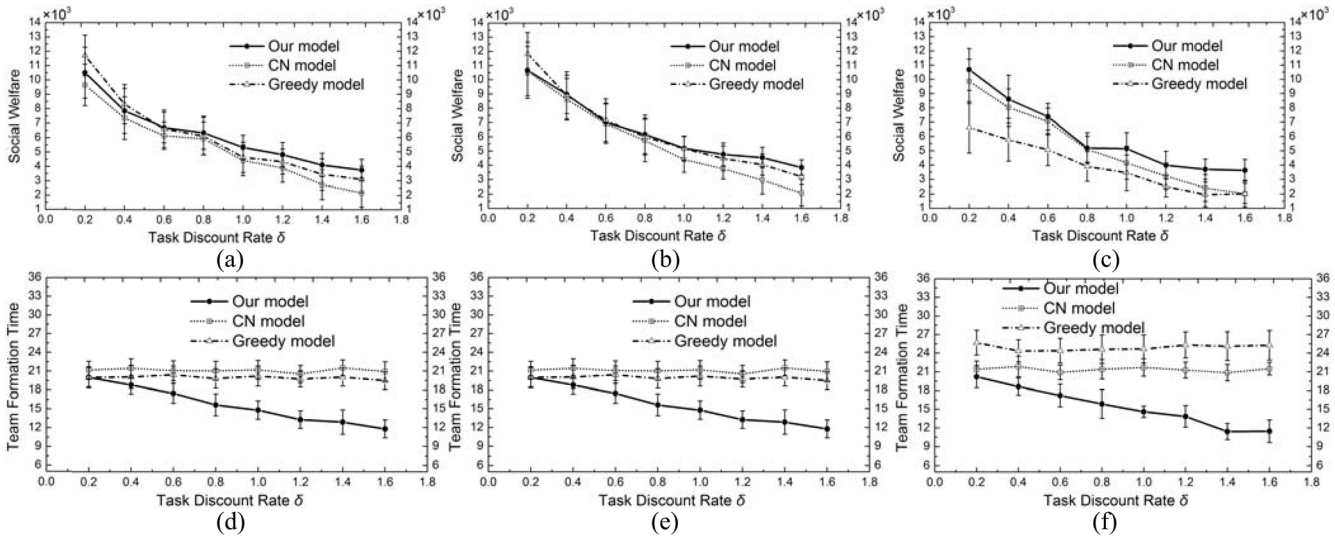

Fig. 2. Effect of task discount rate on different social team formation models' SW (a–c) and team formation time (d–f). (a) and (d) Random. (b) and (e) SMW. (c) and (f) SF.

team formation time in SF than that in Random (or SMW). On one hand, in Greedy, as long as a worker has the skills the team requires, the team manager would negotiate with the worker for skill provision, leading to the manager consume much time on making a contract with a freelancer and breaking a contract with certain existing contractor. And because of the shorter *Dia* and *Aspl*, this repetitive action of making and breaking contracts will happen more frequently, which will consume much more team formation time. While our model negotiates with the worker if and only if he can provide the available skills with cheaper working cost and CN only negotiates the worker that can provide the skills the team current lacks. Therefore, *Dia* and *Aspl* have not much effect on our model and CN model.

3) Within all the networks, as the network degree increases, our model, Greedy and CN will consume much more time on team formation [Fig. 1(d)–(f)]. However, the SWs of our model, Greedy and CN increase gradually. The potential reason is that on one hand, the larger the network degree, the managers will have more social neighbors, and then they will have higher probability to negotiate with the cheap workers, leading to an increase of their profit. On the other hand, the more social neighbors can make these managers access different kinds of skills, thereby increasing the success rate in task completion, which will also increase system SW.

b) Effect of task discount rate on social welfare and team formation time: Fig. 2 shows the SW [Fig. 2(a)–(c)] and team formation time [Fig. 2(d)–(f)] of these models on different task discount rates, where network degree is 8. From Fig. 2, we have the following two findings.

1) Within all the networks, the team formation time of Greedy and CN stay nearly invariable, while surprisingly, our model decreases with the increase of task discount rates [Fig. 2(d)–(f)]. This can be explained as follows: on one hand, in our model, the manager's decision on hiring workers depends on the remaining time for team formation. This means that when the task discount rate becomes larger, the threshold $\text{th} = \text{Ev}(\kappa, \tau)/|R_\kappa|$, which is used to evaluate whether a skill is worth hiring, decreases as well. And this lower threshold will increase the probability of workers to be hired, thereby resulting in a fast team formation. While in CN and Greedy, the team manager's decision on hiring a skill is time independent and when the task discount becomes larger, the team formation time stays invariable.
2) Within all the networks, the SW of our model, CN and Greedy is negatively proportional to task discount rate [Fig. 2(a)–(c)]. This finding is intuitive because the larger the task discount rate, the less task profit will be produced.

B. Comparison With the Benchmark Centralized Optimal Approach

To test the efficiency of our model, we compare our model with the benchmark centralized model [17], where there exists a central controller maintaining information on all of the agents' social connections and working costs. When a task submitted by a requester, the controller can build a team of connected workers that have the least working costs. This is an ideal model, which is impractical, but it can be used as an upper bound of system performance.

1) Experimental Setting: This experimental setting (including the workers and SN) is similar to the setting in Section VI-A, a notable exception is that because the benchmark centralized model does not consume any team formation time, here we set the task discount rate $\delta = 0$ for the correct comparison. Moreover, in this kind of experiment, we only consider the effect of network degree on our model and this centralized model's SW performance.

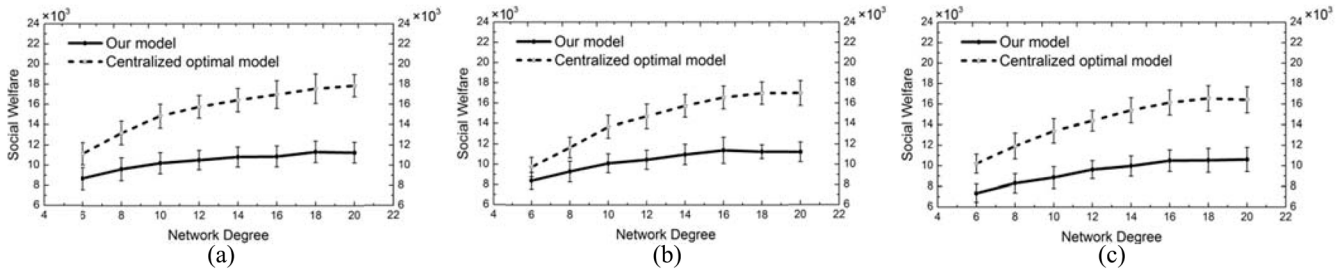

Fig. 3. SW comparison of our model with the centralized optimal model. (a) Random. (b) SMW. (c) SF.

2) Experimental Results: The comparison results are shown in Fig. 3, from which we can observe that: 1) both of the centralized model and our model's SWs increase with the network degree [Fig. 3(a)] and 2) in all of the experiments, although the benchmark centralized optimal model produces more SW than our model's, our model preserves at least 70% of SW of the benchmark centralized optimal model [Fig. 3(b)], which can validate the efficiency of our model's efficiency on producing SW to some extent.

VII. Conclusion

In real-world social team crowdsourcing markets, the requester aims to build a professional and collaborative team for task completion, and the worker aims to join the desirable team for its own profit maximization. Being aware of the dynamics, uncertainty, heterogeneous working cost, team competition and time factor during social team crowdsourcing, we first develop a set of heuristics connecting these interdependent factors indirectly to approximate the requester's and worker's objective functions. Then to satisfy the requester and worker's conflict objectives, we propose a decentralized team formation model for the requester to negotiate with the workers on skill provision and skill provision cost. These negotiation strategies are useful for real-world requesters, since this mechanism closely models real-world crowdsourcing markets (i.e., worker's selfish nature). Theoretic analyses ensure that the requester can always recruit the worker's optimal skill provision that can yield the maximal profit for the requester. Moreover, we also conduct a series of experiment to highlight the efficiency, robustness and scalability of the proposed social team formation model. The experimental results determine that compared to the optimal centralized model, our model can preserve desirable percentage of SW. Moreover, compared to the traditional decentralized approaches, the proposed team formation approach not only builds a feasible (i.e., professional and collaborative) team of inexpensive workers, but also reduces social team formation time.

In this paper, the requesters and workers are assumed to break the partial contract without suffering any penalty. However, in real-life scenarios, breaking a contract unilaterally will always suffer certain penalty such as monetary compensation or reputation loss [26], [42]. Therefore, in the future, it is more practical to consider the decommitment penalty during negotiation and investigate how to optimize the penalty strategy.

References

[1] D. Yang, G. Xue, X. Fang, and J. Tang, "Crowdsourcing to smartphones: Incentive mechanism design for mobile phone sensing," in *Proc. 18th Annu. Int. Conf. Mobile Comput. Netw. (MobiCom)*, Istanbul, Turkey, 2012, pp. 173–184.

[2] G. Goel, A. Nikzad, and A. Singla, "Mechanism design for crowdsourcing markets with heterogeneous tasks," in *Proc. 2nd AAAI Conf. Human Comput. Crowdsourcing (HCOMP)*, Pittsburg, CA, USA, Nov. 2014, pp. 77–86.

[3] L. Tran-Thanh, M. Venanzi, A. Rogers, and N. R. Jennings, "Efficient budget allocation with accuracy guarantees for crowdsourcing classification tasks," in *Proc. 12th Int. Conf. Auton. Agents Multiagent Syst. (AAMAS)*, St. Paul, MN, USA, May 2013, pp. 901–908.

[4] M.-C. Yuen, I. King, and K.-S. Leung, "A survey of crowdsourcing systems," in *Proc. 3rd Int. Conf. Privacy Security Risk Trust Social Comput. (PASSAT/SocialCom)*, Boston, MA, USA, Oct. 2011, pp. 766–773.

[5] J. J. Horton and L. B. Chilton, "The labor economics of paid crowdsourcing," in *Proc. 11th ACM Conf. Elect. Commer. (EC)*, Cambridge, MA, USA, Jun. 2010, pp. 209–218.

[6] L. Tran-Thanh, T. D. Huynh, A. Rosenfeld, S. D. 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. (AAMAS)*, Paris, France, May 2014, pp. 477–484.

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

[8] A. Kittur and R. E. Kraut, "Harnessing the wisdom of crowds in Wikipedia: Quality through coordination," in *Proc. ACM Conf. Comput. Supported Cooperative Work (CSCW)*, San Diego, CA, USA, Nov. 2008, pp. 37–46.

[9] I. Lykourentzouioanna, A. Antoniou, Y. Naudet, and S. P. Dow, "Personality matters: Balancing for personality types leads to better outcomes for crowd teams," in *Proc. 19th ACM Conf. Comput. Supported Cooperative Work Soc. Comput. (CSCW)*, San Francisco, CA, USA, Feb./Mar. 2016, pp. 260–273.

[10] T. Wolf, A. Schröter, D. Damian, L. D. Panjer, and T. H. D. Nguyen, "Mining task-based social networks to explore collaboration in software teams," *IEEE Softw.*, vol. 26, no. 1, pp. 58–66, Jan./Feb. 2009.

[11] J. Chamberlain, "Groupsourcing: Distributed problem solving using social networks," in *Proc. 2nd AAAI Conf. Human Comput. Crowdsourcing (HCOMP)*, Pittsburgh, PA, USA, Nov. 2014, pp. 22–29.

[12] W. Liu *et al.*, "Mining top K spread sources for a specific topic and a given node," *IEEE Trans. Cybern.*, vol. 45, no. 11, pp. 2472–2483, Nov. 2015.

[13] G. Miao *et al.*, "Understanding task-driven information flow in collaborative networks," in *Proc. 21st Int. Conf. World Wide Web (WWW)*, Lyon, France, Apr. 2012, pp. 849–858.

[14] P. De Meo, E. Ferrara, D. Rosaci, and G. M. L. Sarné, "Trust and compactness in social network groups," *IEEE Trans. Cybern.*, vol. 45, no. 2, pp. 205–216, Feb. 2015.

[15] T. Lappas, K. Liu, and E. Terzi, "Finding a team of experts in social networks," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min. (KDD)*, Paris, France, 2009, pp. 467–476.

[16] S. Datta, A. Majumder, and K. V. M. Naidu, "Capacitated team formation problem on social networks," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min. (KDD)*, Beijing, China, Aug. 2012, pp. 1005–1013.

[17] T. Voice, M. Polukarov, and N. R. Jennings, "Coalition structure generation over graphs," *J. Artif. Intell. Res.*, vol. 45, no. 1, pp. 165–196, 2012.

[18] S. Liemhetcharat and M. Veloso, "Weighted synergy graphs for effective team formation with heterogeneous ad hoc agents," *Artif. Intell.*, vol. 208, pp. 41–65, Mar. 2014.

[19] L. Sabattini, C. Secchi, and N. Chopra, "Decentralized estimation and control for preserving the strong connectivity of directed graphs," *IEEE Trans. Cybern.*, vol. 45, no. 10, pp. 2273–2286, Oct. 2015.

[20] A. Anagnostopoulos, L. Becchetti, C. Castillo, A. Gionis, and S. Leonardi, "Online team formation in social networks," in *Proc. 21st Int. Conf. World Wide Web (WWW)*, Lyon, France, Apr. 2012, pp. 839–848.

[21] C.-T. Li, M.-K. Shan, and S.-D. Lin, "On team formation with expertise query in collaborative social networks," *Knowl. Inf. Syst.*, vol. 42, no. 2, pp. 441–463, 2015.

[22] M. Kargar, A. An, and M. Zihayat, "Efficient bi-objective team formation in social networks," in *Proc. Eur. Conf. Mach. Learn. Principles Pract. Knowl. Disc. Databases (ECML-PKDD)*, Bristol, U.K., Sep. 2012, pp. 483–498.

[23] S. S. Rangapuram, T. Bühler, and M. Hein, "Towards realistic team formation in social networks based on densest subgraphs," in *Proc. 22nd Int. Conf. World Wide Web (WWW)*, Rio de Janeiro, Brazil, 2013, pp. 1077–1088.

[24] Y. Jiang and J. C. Jiang, "Understanding social networks from a multiagent perspective," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 10, pp. 2743–2759, Oct. 2014.

[25] M. Kokkodis, P. Papadimitriou, and P. G. Ipeirotis, "Hiring behavior models for online labor markets," in *Proc. 8th ACM Int. Conf. Web Search Data Min. (WSDM)*, Shanghai, China, 2015, pp. 223–232.

[26] B. An, V. Lesser, D. Irwin, and M. Zink, "Automated negotiation with decommitment for dynamic resource allocation in cloud computing," in *Proc. 9th Int. Conf. Auton. Agents Multiagent Syst. (AAMAS)*, Toronto, ON, Canada, May 2010, pp. 981–988.

[27] W. Wang and Y. Jiang, "A practical negotiation-based team formation model for non-cooperative social networks," in *Proc. IEEE Int. Conf. Tools Artif. Intell. (ICTAI)*, Limassol, Cyprus, Nov. 2014, pp. 344–351.

[28] D. Ye, M. Zhang, and D. Sutanto, "Self-adaptation-based dynamic coalition formation in a distributed agent network: A mechanism and a brief survey," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 5, pp. 1042–1051, May 2013.

[29] L. Niu, F. Ren, M. Zhang, and Q. Bai, "A concurrent multiple negotiation protocol based on colored Petri nets," *IEEE Trans. Cybern.*, to be published, doi: 10.1109/TCYB.2016.2577635.

[30] Q. Bao, W. K. Cheung, Y. Zhang, and J. Liu, "A component-based diffusion model with structural diversity for social networks," *IEEE Trans. Cybern.*, to be published, doi: 10.1109/TCYB.2016.2537366.

[31] M. M. de Weerdt, Y. Zhang, and T. Klos, "Multiagent task allocation in social networks," *Auton. Agents Multi Agent Syst.*, vol. 25, no. 1, pp. 46–86, 2012.

[32] A. Azaria, Y. Aumann, and S. Kraus, "Automated agents for reward determination for human work in crowdsourcing applications," *Auton. Agents Multi Agent Syst.*, vol. 28, no. 6, pp. 934–955, 2014.

[33] Y. Singer and M. Mittal, "Pricing mechanisms for crowdsourcing markets," in *Proc. 22nd Int. Conf. World Wide Web (WWW)*, Rio de Janeiro, Brazil, May 2013, pp. 1157–1166.

[34] L. Tran-Thanh, S. Stein, A. Rogers, and N. R. Jennings, "Efficient crowdsourcing of unknown experts using bounded multi-armed bandits," *Artif. Intell.*, vol. 214, pp. 89–111, Sep. 2014.

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

[36] A. W. Woolley, C. F. Chabris, A. Pentland, N. Hashmi, and T. W. Malone, "Evidence for a collective intelligence factor in the performance of human groups," *Science*, vol. 330, no. 6004, pp. 686–688, 2010.

[37] S. P. Dow, A. Kulkarni, S. R. Klemmer, and B. Hartmann, "Shepherding the crowd yields better work," in *Proc. 26th Conf. Comput. Supported Cooperative Work (CSCW)*, Seattle, WA, USA, Feb. 2012, pp. 1013–1022.

[38] M. Jamali and M. Ester, "A transitivity aware matrix factorization model for recommendation in social networks," in *Proc. 22nd Int. Joint Conf. Artif. Intell. (IJCAI)*, Barcelona, Spain, Jul. 2011, pp. 2644–2649.

[39] R. Forsati, M. Mahdavi, M. Shamsfard, and M. Sarwat, "Matrix factorization with explicit trust and distrust side information for improved social recommendation," *ACM Trans. Inf. Syst.*, vol. 32, no. 4, pp. 1–38, 2014.

[40] B. Fu, G. Xu, L. Cao, Z. Wang, and Z. Wu, "Coupling multiple views of relations for recommendation. Advances in knowledge discovery and data mining," in *Proc. 19th Pac. Asia Conf. Knowl. Disc. Data Min. (PAKDD)*, May 2015, pp. 732–743.

[41] Y. Jiang and J. Jiang, "Contextual resource negotiation-based task allocation and load balancing in complex software systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 20, no. 5, pp. 641–653, May 2009.

[42] Y. Jiang, Y. Zhou, and W. Wang, "Task allocation for undependable multiagent systems in social networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 8, pp. 1671–1681, Aug. 2013.

[43] W. Wang and Y. Jiang, "Multiagent-based allocation of complex tasks in social networks," *IEEE Trans. Emerg. Topics Comput.*, vol. 3, no. 4, pp. 571–584, Dec. 2015.

[44] M. E. Gaston and M. DesJardins, "The effect of network structure on dynamic team formation in multi-agent systems," *Comput. Intell.*, vol. 24, no. 2, pp. 122–157, 2008.

[45] M. E. Gaston and M. desJardins, "Agent-organized networks for dynamic team formation," in *Proc. 4th Int. Conf. Auton. Agents Multiagent Syst. (AAMAS)*, Utrecht, The Netherlands, Jul. 2005, pp. 230–237.

[46] L. Sless, N. Hazon, S. Kraus, and M. Wooldridge, "Forming coalitions and facilitating relationships for completing tasks in social networks," in *Proc. 13th Int. Conf. Auton. Agents Multiagent Syst. (AAMAS)*, Paris, France, 2014, pp. 261–268.

[47] A. Peleteiro, J. C. Burguillo, and S. Y. Chong, "Exploring indirect reciprocity in complex networks using coalitions and rewiring," in *Proc. 13th Int. Conf. Auton. Agents Multiagent Syst. (AAMAS)*, Paris, France, May 2014, pp. 669–676.

[48] Y. Jiang and J. C. Jiang, "Diffusion in social networks: A multiagent perspective," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 45, no. 2, pp. 198–213, Feb. 2015.

[49] W. Wang, Z. He, P. Shi, W. Wu, and Y. Jiang, "Truthful team formation for crowdsourcing in social networks," in *Proc. 15th Int. Conf. Auton. Agents Multiagent Syst. (AAMAS)*, Singapore, May 2016, pp. 1327–1328.

[50] F. Bergenti, E. Franchi, and A. Poggi, "Selected models for agent-based simulation of social networks," in *Proc. 3rd Symp. Social Netw. Multiagent Syst. (SNAMAS)*, York, U.K., Apr. 2011, pp. 27–32.

Wanyuan Wang received the Ph.D. degree in computer science from Southeast University, Nanjing, China, in 2016.

He is currently a Post-Doctoral Researcher with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. He has published several articles in refereed journals and conference proceedings, such as the IEEE TRANSACTIONS, the International Conference on Autonomous Agents and Multiagent Systems, and the IEEE International Conference on Tools With Artificial Intelligence (ICTAI). His current research interests include social networks, multiagent systems, and crowdsourcing.

Mr. Wang was a recipient of the Best Student Paper Award from the ICTAI'14.

Jiuchuan Jiang received the M.E. degree in computer science from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2009.

He has published several scientific articles in refereed journals and conference proceedings. His current research interests include multiagent systems, crowdsourcing, and social networks.

Bo An received the Ph.D. degree in computer science from the University of Massachusetts at Amherst, Amherst, MA, USA.

He is a Nanyang Assistant Professor with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. He was a Post-Doctoral Researcher with the University of Southern California, Los Angeles, CA, USA. He has published over 50 referred papers at AAMAS, IJCAI, AAAI, ICAPS, KDD, JAAMAS, and IEEE TRANSACTIONS. His current research interests include artificial intelligence, multiagent systems, game theory, and optimization.

Mr. An was a recipient of the 2010 International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS) Victor Lesser Distinguished Dissertation Award, the Operational Excellence Award from the Commander, the First Coast Guard District of the U.S., the Best Innovative Application Paper Award at the 11th International Joint Conference on Autonomous Agents and Multi-Agent Systems, the Deployed Innovative Application Award at the 28th Annual Conference on Innovative Applications of Artificial Intelligence, and the 2012 INFORMS Daniel H. Wagner Prize for Excellence in Operations Research Practice. He is an Editorial Board Member of the *Journal of Artificial Intelligence Research* and an Associate Editor of the *Journal of Autonomous Agents and Multi-Agent Systems*. He was elected to the board of directors of IFAAMAS.

Yichuan Jiang (SM'13) received the Ph.D. degree in computer science from Fudan University, Shanghai, China, in 2005.

He is currently a Full Professor with the Distributed Intelligence and Social Computing Laboratory, School of Computer Science and Engineering, Southeast University, Nanjing, China. He has published over 80 scientific articles in refereed journals and conference proceedings, such as the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART A: SYSTEMS AND HUMANS, the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART C: APPLICATIONS AND REVIEWS, the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS, the IEEE TRANSACTIONS ON CYBERNETICS, *ACM Transactions on Autonomous and Adaptive Systems*, the *Journal of Autonomous Agents and Multi-Agent Systems*, the *Journal of Parallel and Distributed Computing*, the International Joint Conference on Artificial Intelligence, the International Conference on Autonomous Agents and Multiagent Systems, and the IEEE International Conference on Tools With Artificial Intelligence (ICTAI). His current research interests include multiagent systems, social networks, and social computing.

Prof. Jiang was a recipient of the Best Paper Award and the Best Student Paper Award from PRIMA and ICTAI.

Bing Chen received the B.S. and M.S. degrees in computer engineering from the Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing, China, in 1992 and 1995, respectively, and the Ph.D. degree from the College of Information Science and Technology, NUAA, in 2008.

Since 1998, he has been with the NUAA, where he is currently a Professor with the School of Computer Science and Technology. His current research interests include computer networks, wireless communications, and social networks.