Received 28 June 2014; revised 19 October 2014; accepted 7 February 2015. Date of publication 00 xxxx 0000;
date of current version 00 xxxx 0000.

Digital Object Identi er 10.1109/TETC.2015.2403200

Multiagent-Based Allocation of Complex Tasks in Social Networks

WANYUAN WANG, (Student Member, IEEE), AND YICHUAN JIANG, (Senior Member, IEEE)

School of Computer Science and Engineering, Southeast University, Nanjing 211189, China
Key Laboratory of Computer Network and Information Integration, Ministry of Education,
Southeast University, Nanjing 211189, China

CORRESPONDING AUTHOR: Y. JIANG (yjiang@seu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61170164 and Grant 61472079, 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 Program for Distinguished Talents of Six Domains in Jiangsu Province under Grant 2011-DZ023.

ABSTRACT In many social networks (SNs), social individuals often need to work together to accomplish a complex task (e.g., software product development). In the context of SNs, due to the presence of social connections, complex task allocation must achieve satisfactory social effectiveness; in other words, each complex task should be allocated to socially close individuals to enable them to communicate and collaborate effectively. Although several approaches have been proposed to tackle this so-called social task allocation problem, they either suffer from being centralized or ignore the objective of maximizing the social effectiveness. In this paper, we present a distributed multiagent-based task allocation model by dispatching a mobile and cooperative agent to each subtask of each complex task, which also addresses the objective of social effectiveness maximization. With respect to mobility, each agent can transport itself to a suitable individual that has the relevant capability. With respect to cooperativeness, agents can cooperate with each other by forming teams and moving to a suitable individual jointly if the cooperation is bene cial. Our theoretical analyses provide provable performance guarantees of this model. We also apply this model in a set of static and dynamic network settings to investigate its effectiveness, scalability, and robustness. Through experimental results, our model is determined to be effective in improving the system load balance and social effectiveness; this model is scalable in reducing the computation time and is robust in adapting the system dynamics.

INDEX TERMS Complex task allocation, social networks, multiagent, social effectiveness, load balancing.

I. INTRODUCTION

Today's many online social networks (SNs) [1], such as LinkedIn [2] and GitHub [3], provide a good marketing platform for enterprises, organizations or individuals conducting business. Through social network platforms, the enterprises (organizations or individuals) post their tasks to all users and recruit a set of professional users to accomplish their tasks. In this so-called social task allocation problem, we especially focus on the cases where tasks are complex. The complex tasks differ from common tasks in the sense that each complex task consists of a set of interdependent subtasks that require coordination with one another. In the context of SNs, the success of completing a complex task depends not only on how professional the recruited users are [4], [5] and how many workloads these recruited users undertake [6], [7], but also on how effectively they can communicate to perform the interdependent subtasks [8] [10].

We can consider the following scenario as a motivating example. An IT manager in LinkedIn wants to recruit a team of software engineers that can meet the skill requirements of a complex software product *P*, which includes six activities {*Requirement Analysis* (*RA*), *Architecture Design* (*AD*), *Implementation* (*IM*), *Testing* (*TE*), *Deployment* (*DE*), *Maintenance* (*MA*)}. During software development, collaboration among the developers is often required for accomplishing the interdependent activities [11]. The interdependent relationships among these activities are shown in Fig. 1(a), where an edge between two activities indicates that the engineers who perform

quickly, making it a desirable option for dynamic large-scale applications.

The remainder of this paper is organized as follows. In the next section, we give the de nition of the social complex task allocation problem and its objective; in Section 3, we propose the multiagent-based task allocation models with non-cooperative and cooperative agents. In Section 4, we analyze our model's properties, and Section 5 conducts a set of experiments to evaluate our model's effectiveness, scalability and robustness. In Section 6, we provide a brief review of related work on task allocation among social network subjects. Finally, we present our paper's conclusions and discuss future work in Section 7.

II. PROBLEM DESCRIPTION

A. NOTATIONS

We consider the social complex task allocation problem consisting of a social network SN and a set of complex tasks $0=\{CT_1,CT_2;::::;CT_n\}$. The social network $SN=<N,E>$ is an undirected graph, where $N=\{n_1,n_2;::::;n_m\}$ is the set of nodes (hereafter, we use the terms ``node,'' ``user'' and ``individual'' interchangeably) and $\forall(n_i;n_j)\in E$ indicates the existence of a connection between nodes n_i and n_j. The connections among the nodes represent communication possibilities. Let there be l types of available capabilities of nodes $C=\{c_1;c_2;::::;c_l\}$ in a SN. Each node $n_i\in N$ owns some capabilities $O_{ni}\subseteq C$, which make it eligible to perform the subtask that requires the capability $c_j\in O_{ni}$.

A complex task is a task that can be divided into several subtasks that are dependent on one another [24]. Each complex task $CT_i\in 0$ then can be represented by an undirected graph $<TV, TE>$, where $TV=\{t_{i1},t_{i2};::::;t_{ik}\}$ is the set of subtasks of CT_i, and $\forall(t_{ip};t_{iq})\in TE$ indicates that t_{ip} and t_{iq} are interdependent on one another.[1] Each subtask $t_{ij}\in TV$ then can be de ned by a 3-tuple $<R(t_{ij}),\bullet(t_{ij});n(t_{ij})>$, where $R(t_{ij})\subseteq C$ is the capability that the subtask t_{ij} requires (for simplicity, we assume that each subtask requires only a single capability, i.e., $|R(t_{ij})|=1$). The interdependent subtask set $\bullet(t_{ij})=\{t_{ik}|(t_{ij};t_{ik})\in TE\}$ represents the relevant subtasks that t_{ij} must coordinate with. The node location function $n(t_{ij}):TV\rightarrow N$ indicates the node on which t_{ij} is allocated. A ***valid task allocation*** δ is de ned as the mapping of subtask $\forall t_{ij}\in TV$ to a node n_k such that n_k has the capability to perform t_{ij} (i.e., $R(t_{ij})\in O_{nk}$).

B. OBJECTIVES

1) LOAD BALANCING

As discussed earlier, one of the main objectives of social complex task allocation is load balancing, i.e., assign the subtasks among the nodes as evenly as possible such that no subtask must wait very much time to be executed. There are many measures to quantify the load balance extent, such as the maximum load over all of the nodes [7] and the standard deviation of nodes' loads [22]. However, to re ect the advantage of our muliagent-based task allocation model (presented in Section 3), we adopt an alternative load balance measure, which can be called the *social waiting cost*.

De nition 1 (Social Waiting Cost): Given a valid allocation δ of subtask $\forall t_{ij}\in CT_i$ ($CT_i\in 0$) to a node $n_k\in N$, the social waiting cost of all of the subtasks, $SWC(\delta)$, is de ned as follows:

$$SWC(\delta)=\sum_{n_i\in N} L_{n_i}(L_{n_i}+1)/2 \quad (1)$$

where $L_{ni}=|\{t_{jk}|n(t_{jk})=n_i\}|$ is the number of subtasks that are allocated on node n_i. For a given node n_i, let the number of subtasks that are allocated on n_i be L_{ni}; then, the rst subtask must wait one unit of computation time to be completed, the second subtask must wait two units of computation time, and inductively, the L_{ni}th subtask must wait L_{ni} units of time. The total waiting cost of the subtasks allocated on node n_i then is $\sum_{1\le i\le L_{n_i}} i=L_{n_i}(L_{n_i}+1)/2$. This social waiting cost de nition is motivated by [23], which is powerful enough to quantify the load balance extent.

Property 1: Given a social task allocation problem, the smaller the social waiting cost of a valid allocation δ, the more balanceable the allocation δ is.

2) SOCIAL EFFECTIVENESS

In addition to aiming at a fair allocation of subtasks among the nodes, these allocated nodes should also communicate with one another effectively in such a way that they can complete a complex task successfully [18]. Given any two nodes n_i and n_j, however, it is not easy for a task manager to determine whether they can communicate effectively if this manager does not know them well. In many SNs, the connection always represent a positive social relationship between social individuals such as friendship in acquaintance networks [16], partnership in collaboration networks [5] or location proximity in opportunistic mobile networks [17]. Therefore, social distance can be used as a good indicator of social effectiveness [2], [3], [9], [10]. The social distance between nodes n_i and n_j, $d(n_i;n_j)$ is the sum of the connections on the shortest path that connects the two nodes, and the shorter the distance between them, the more effectively they can communicate (or equivalently, the fewer the communication cost will be incurred), and vice versa. In the following, we use a simple but intuitive and reasonable *social communication cost* measure to quantify the social effectiveness.

De nition 2 (Social Communication Cost): Given a valid allocation δ of subtask $\forall t_{ij}\in CT_i$ ($CT_i\in 0$) to a node $n_k\in N$, the social communication cost of all subtasks, $SCC(\delta)$, is de ned as follows:

$$SCC(\delta)=\sum_{CT_i\in 0}\sum_{t_{ij}\in CT_i}\sum_{t_{ik}\in\bullet(t_{ij})} d(n(t_{ij});n(t_{ik}))/2 \quad (2)$$

[1] In this paper, we assume that there are no rigorous precedence orders among the interdependent subtasks, which is reasonable in real-world applications. For example, during the development of a software product, although the *Implementation* activity occurs prior to the *Testing* activity, the *Implementation* engineer must also wait for the *Testing* engineer's feedback for debugging and optimization.

Property 2: Given a social task allocation problem, the smaller the social communication cost of a valid allocation δ, the more socially effective the allocation δ is.

C. TRADE-OFF BETWEEN LOAD BALANCING AND SOCIAL EFFECTIVENESS

We are mainly concerned with allocating the subtasks to nodes with the aims of both load balancing and social effectiveness maximization, which is a bi-objective optimization problem, and the two objectives are often con icting. A typical way to solve the bi-objective optimization problem is to transform the problem into a single objective problem [12], [14]. In this paper, we adopt this idea by combining the two objective functions (i.e., social waiting cost and social communication cost) into a single objective function (i.e., the social execution cost).

De nition 3 (Social Execution Cost): Given a valid allocation δ of subtask $\forall t_{ij} \in CT_i$ ($CT_i \in 0$) to a node $n_k \in N$, the social execution cost of all subtasks, $SEC(\delta)$, is de ned as follows: $SEC(\delta) = SWC(\delta) + SCC(\delta)$.

The coef cients and (; >0) determine the in uences of their corresponding terms and are application dependent. For example, in load-oriented scenarios such as proposal evaluation, the communication cost might not be signi cant compared to the waiting cost [7]. However, in communication-oriented applications such as developing software products, engineers must spend a substantial amount of effort on communication [2].

Finally, the social complex task allocation problem that is studied in this paper can be de ned as follows:

De nition 4 (Social Complex Task Allocation Problem): Given a social network $SN=<N; E>$ and a nite set of complex tasks 0 , the social complex task allocation problem is to determine the optimal valid allocation δ that has the minimum social execution cost, i.e.,

$$\min SEC(\delta)$$
$$s.t.\ R(t_{ij}) \in O_{n(t_{ij})}; \quad \forall t_{ij} \in CT_i; CT_i \in 0$$

Property 3: Social complex task allocation problem is NP-hard. The hardness proof follows directly from the traditional known NP-hard social team formation problem discussed in [2], which is a specialization of this social complex task allocation problem when the load balancing objective is ignored (i.e., =0).

Because our problem cannot be solved optimally within polynomial time unless P=NP, in this paper, we propose an ef cient distributed social complex task allocation model by utilizing the multiagent technology, which also provides a provable performance guarantee. Multiagent technologies have been employed widely for distributed problem solving [1], [12] [15], [19], [21] [31]. Motivated by the studies [22] [24], in our model, we use the mobile and cooperative agent to carry the subtask to search for the suitable target node.

III. THE MODEL

We formulate the multiagent-based social complex task allocation model as follows. First, we dispatch an agent a_{i+j} for each subtask $t_{ij} \in CT_i$ ($A=\{a_1, a_2, \ldots, a_n\}$ denotes the collection of agents in the system). Each agent $a_i \in A$ then can be de ned by a 2-tuple $<ST(a_i), IA(a_i)>$, where $ST(a_i)$ indicates the subtask that it carries, and the interdependent agent set $IA(a_i)=\{a_p, \ldots, a_q\}$ represents those agents whose subtasks have direct dependencies with the subtask of a_i, i.e., $IA(a_i) = \{a_j | ST(a_j) \in \bullet(ST(a_i))\}$. Second, we model each agent with the mobility property. With respect to mobility, we mean that each agent a_i can transport itself to a ***suitable node*** that owns the capability to perform a_i. Given the mobility property of the agents, if there are multiple nodes suitable for agent a_i, which node should a_i choose to queue at? To answer this question, in Section III-A, we rst investigate the non-cooperative setting where each agent is tempted to move to the most suitable node from its own perspective and elaborate why the agents should be endowed with a cooperative property. Then, in Section III-B, we develop an ef cient multiagent-based task allocation model with cooperative agents.

A. THE MODEL WITH NON-COOPERATIVE AGENTS

In the non-cooperative setting, each agent attempts to queue at the optimal suitable node that produces the minimum execution cost for itself. The execution cost of an agent consists of the waiting cost and the communication cost. Denote by n_{ai} the node that agent a_i queues at, and the waiting, communication and execution costs of a_i are de ned as follows:

De nition 5 (Waiting Cost of an Agent): The agent a_i's cost of waiting to be completed by node n_{ai}, $Wc(a_i; n_{ai})$, is given by the total number of agents that queue at n_{ai} ,i.e., $Wc(a_i; n_{ai})=|\{a_j | n_{aj}=n_{ai}\}|$.

De nition 6 (Communication Cost of an Agent): The agent a_i's cost of communicating with all of its interdependent agents $IA(a_i)$ is given by $Cc(a_i; n_{ai})=\sum_{a_j \in IA(a_i)} d(n_{a_i}; n_{a_j})$:

From the viewpoint of a_i, this agent can be considered successfully executed if it has waited its turn in the queue of n_{ai} and has communicated with its interdependent agents $IA(a_i)$.

De nition 7 (Execution Cost of an Agent): Assume that the waiting and communication costs of agent a_i are $Wc(a_i; n_{ai})$ and $Cc(a_i; n_{ai})$, respectively. Then, the execution cost of a_i is $Ec(a_i; n_{ai}) = Wc(a_i; n_{ai}) + Cc(a_i; n_{ai})$.

The meanings of the two coef cients and are similar to those described in De nition 3. This execution cost de nition has many desirable properties that satisfy the objective of the social complex task allocation problem.

Property 4: An agent prefers to queue at a node that has a small agent load.

This agent's preference reduces the social waiting cost.

Property 5: An agent prefers to queue at a node at which its interdependent agents reside.

FIGURE 2. A simple social complex task allocation instance. (a) The network. (b) The complex task. (c) The equilibrium solution S=$\{n_2, n_2, n_2, n_2\}$. (d) The optimal solution $S*$=$\{n_1, n_1, n_1, n_2\}$.

Given this preference, in the case that all of the interdependent agents of a_i queue at n_j, a_i is more likely to queue at n_j because of the zero intra-node communication cost, resulting in a reduced social communication cost.

Formally, in the dynamic muliagent model, each agent a_i's strategy s_i is the node that it selects to queue at (i.e., $s_i \in N$). The strategy set $S=\{s_1,s_2;::::;s_n\}$ of all agents is called the strategy pro le. In the non-cooperative setting, a strategy pro le is in equilibrium if and only if no agent has any incentive to change its strategy (move from its current node to another suitable node) unilaterally, i.e., $\forall a_i \in A$ and $s_i' \neq s_i$, $Ec(a_i; s_i', S_{-i}) \geq Ec(a_i; s_i; S_{-i})$; where $(a_i; s_i', S_{-i})$ is the alternative strategy pro le that is generated only when a_i changes its strategy from s_i to s_i'. If agents search for suitable nodes in a purely sel sh manner, the system can always converge to an equilibrium solution **(later in Section 3.2, we will provide a rigorous proof of the convergence of a multiagent model with the cooperative agents, which encompasses the case where the agents are non-cooperative; thus, we omit this proof here**). However, at the equilibrium state, the social execution cost (*SEC*) produced by the sel sh agents is not necessarily the optimal. To illustrate how bad sel sh behavior is, consider a simple social complex task allocation instance shown in Fig. 2.

Example 1: In Fig. 2(a), there is a network that consists of two interconnected nodes, n_1 and n_2. Node n_1 holds the capability set $\{c_1,c_2,c_3\}$, and n_2 holds the capability set $\{c_1,c_2,c_3,c_4\}$. Now, assume that a complex task CT is submitted to the system; this task comprises four interdependent subtasks $t_1=\{c_1\}$, $t_2=\{c_2\}$, $t_3=\{c_3\}$ and $t_4=\{c_4\}$ (Fig. 2(b)). The coef cients and , involved in the calculation of the execution cost, are set to 3 and 5. According to the multiagent model, four mobile agents $a_1=<t_1,\{a_2,a_3\}>$, $a_2=<t_2,\{a_1,a_3\}>$, $a_3=<t_3,\{a_1,a_2;a_4\}>$, and $a_4=<t_4,\{a_3\}>$ are dispatched to the subtasks $\{t_i|1\leq i\leq 4\}$, respectively. Next, we consider the strategy pro le $S=\{n_2,n_2,n_2,n_2\}$, i.e., a_1, a_2, a_3 and a_4 all queue at node n_2. At pro le S, from each agent's own viewpoint, they are queuing at the optimal suitable node: $\forall a_i (i=1;2)$, $Ec(a_i; n_2, S_{-i})= L_{n2} + d(n_2; n_{IA}(ai))=4 \ =12<Ec(a_i; n_1; S_{-i})= \ +2 \ =13$; for a_3, $Ec(a_3,n_2,S_{-i})=4 \ =12<Ec(a_3,n_1,S_{-i})= \ +3 \ = \ 18$, and for a_4, node n_2 is the only suitable node that has the capability c_4. Thus, the strategy pro le S is an equilibrium solution that has $\frac{1}{2} \sum_{1\leq i\leq 2} L_{n_i}(L_{n_i} + 1) + \frac{1}{2} \sum_{1\leq i\leq 4} d(n_{a_i}; n_{IA(a_i)})=10 \ =30$ unit *SECs*. However, the optimal solution of this instance is $S*=\{n_1,n_1,n_1,n_2\}$ (i.e., a_1, a_2 and a_3 queue at node n_1, and a_4 queues at n_2), which only produces 7 $+$ $=26$ unit *SECs*. It is worthwhile noting that this optimal solution $S*$ can be easily achieved from strategy pro le S if the agents a_1, a_2 and a_3 cooperate with each other by moving from node n_2 to n_1 jointly.

Therefore, it is very bene cial to model the agents being cooperative. The cooperation mechanism employed in this study extends the team formation mechanism [30] by allowing agents to cooperate with each other to form a team and to move the same node jointly. Furthermore, to avoid the cooperation mechanism producing large inter-node message delivering overhead, we constrain each agent a_i in such a way that it can cooperate only with its intra-node agents (i.e., the agents that queue at the same node with a_i's). The network traf c overhead produced by the intra-node negotiation is so small that it can be neglected [24].

B. THE MODEL WITH COOPERATIVE AGENTS

The main idea of the cooperation mechanism implemented by the agents can be brie y described as follows: each agent negotiates with its intra-node agents and decides to cooperate with them by forming a team if cooperation results in a reduced execution cost for the team. The execution cost of an agent team is de ned as follows:

De nition 8 (Execution Cost of a Team): Denote by n_G the suitable node that an agent team G queues at[2]*; then, the team G`s execution cost $Ec(G, n_G)$ is the following*:

$$Ec(G; n_G) = \sum_{1\leq i\leq l_G} (L_{n_G} - l_G + i) + \sum_{a_i\in G} \sum_{a_j\in IA(a_i)} d(n_G; n_{a_j}) \quad (3)$$

The rst term represents the total waiting cost of the team G, where $L_{nG}=|\{a_j|n_{aj}=n_G\}|$ is the number of agents that queue at node n_G, and $l_G=|\{a_j|a_j\in G\}|$ is the number of agents in G. For a given node n_G with the agent load L_{nG}, the rst agent in G must wait $L_{nG} - l_G + 1$ unit cost, the second agent in G must wait $L_{nG} - l_G + 2$ unit cost, and inductively, the l_Gth agent in G requires L_{nG} units of waiting cost. The total waiting cost of the team G queuing at n_G, then, is $\sum_{1\leq i\leq l_G} (L_{n_G} - l_G + i)$. The second term represents the total communication cost of G. **It is worthwhile noting that when there is only one**

[2]A node n_G is suitable for an agent team G if and only if it has the capabilities required by all of the agents in G, i.e., $\forall a_i \in G$, $R(ST(a_i)) \in On_G$.

agent a_i in G, the execution cost of a_i can be recovered in accordance with that de nition 7.

Next, we will illustrate how the team can be formed and the advantage of the team formation protocol. Recall Example 1, at the equilibrium strategy pro le $S=\{n_2,n_2, n_2,n_2\}$; from its own viewpoint, agent a_1 realizes that it is queuing at the optimal node (i.e., node n_2). However, according to the team execution cost de nition, a_1 nds that forming team $G=a_1\cup\{a_2; a_3\}$ with a_2, a_3 and moving to node n_1 jointly produces the less team execution cost: before moving, the team execution cost of G; $Ec(G; n_2)= \sum_{i=1;2;3}(L_{n_G}-I_G+i)+ \sum_{a_i\in G}\sum_{a_j\in IA(a_i)} d(n_G; n_{a_j})=9 =27$, after the team G moves to n_1, the execution cost of Gbecomes $Ec(G,n_1)=6 + =23<27=Ec(G,n_2)$. Then, a_1 will negotiate with a_2 and a_3 for team formation, and because of the cooperative property, agents a_2 and a_3 are willing to join this team, whereby the team is formed.

An agent team (**which also encompasses the case in which there is only one agent in the team, i.e., the non-cooperative case**) prefers to change strategies (i.e., move from its current node to another suitable node) if the strategy changing can reduce the execution cost of the team that it forms. Here, we use the measure of bene t to quantify how much an agent team gains by changing its strategy. The bene t that a team G gains by moving from the suitable node n_x to another suitable node n_y is

$$B(G; n_x; n_y) = Ec(G; n_x) - Ec(G; n_y) \quad (4)$$

Given this bene t de nition, it can be observed that there is no incentive for an agent team G to form a new team $G^*=G\cup\{a_j\}$ by merging the agent a_j that has no dependencies with the agents in G (i.e., $a_j \in \cup_{a_i\in G} IA(a_i)$) because forming such a new team G^* would not decrease any communication cost of the original team$G$ but only increases the waiting cost of G.

Property 6: For any agent $a_i\in A$, it is only bene cial for a_i to form a team with its interdependent agents and interdependent agents' interdependent agents, if necessary.

Therefore, each agent a_i's cooperation domain (a_i) can be further limited within its intra-node interdependent agents and interdependent agents' interdependent agents, if necessary, which can be denoted by $(a_i)=\{a_j|n_{aj}= n_{ai}\wedge ST(a_j)\cdot CT=ST(a_i)\cdot CT\}$ ($ST(a_i)\cdot CT$ means the complex task that subtask $ST(a_i)$ belongs to). Nevertheless, for each agent, even nding the optimal team that yields the largest bene t within its cooperation domain agents is not easy. Assume that agent a_i is queuing at a certain node; identifying the optimal team from its cooperation domain (a_i) must consider the exponential number $O(2^{|(ai)|})$ of possible combinations. To deal with this computationally costly optimization problem, we propose an ef cient *Breadth-First* negotiation mechanism, where each agent forms a bene cial team by negotiating from its intra-node direct interdependent agents to far-away interdependent agents' interdependent agents gradually. To illustrate the negotiation protocol, consider Example 1 again (see Fig. 3).

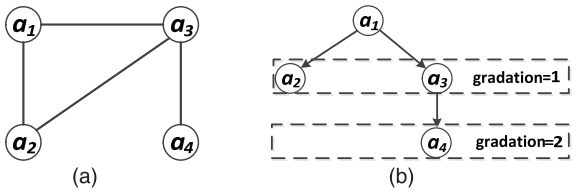

FIGURE 3. (a) The dependency relationships of the agents that queue at node n_2. (b) The negotiation process of agent a_1.

Example 1 (Continue): Suppose that the four interdependent agents $\{a_i|1\le i\le 4\}$ are now queuing at node n_2 (the dependencies of these agents are shown in Fig. 3(a)). Without loss of generality, assume that agent a_1 wants to change strategy. The negotiation process employed by a_1 to form a bene cial team then can be described as in Fig. 3(b): rst, a_1 negotiates with its direct interdependent agents a_2 and a_3, i.e., the agents in gradation 1. If a_1 nds that it is bene cial to form a team with a_2 by moving to a certain node (e.g., n_1) jointly, a_1 will negotiate with a_2 to join this team and proceeds to negotiate with the other agents in gradation 1 (i.e., a_3). Otherwise, if the formation of a team with agent a_2 does not yield any bene t, then a_2 will be removed from the current team. After a_1 has negotiated with all of the agents in gradation 1 and nds that forming a team with a_2 and a_3 (denote a_2 and a_3 as the new joining agents) is bene cial, a_1 2

Algorithm 1: Multiagent-Based Social Complex Task Allocation Model

Repeat the following procedure until no agent team can bene t by changing its strategy.
Pick an agent $a_i \in A$ randomly.
Initialize the ag f_j of agent a_j queuing at n_{ai} to 0.
Initialize $G = \emptyset$, max=0 and $target$=Null.
Create Queue(Q).
Insert Queue(Q; a_i), and set f_i=1.
While $(Q \neq \emptyset)$ **do**
  Set a_x = **Out** Queue(Q) and tag=*false*.
  For $\forall n_j \in N$ **do**
    If $B(G \cup \{a_x\}, n_{ai}; n_j) >$ max && $\forall a_y \in G \cup \{a_x\}, R(ST(a_y)) \in O_{nj}$ **then**
      tag=*true*, $max = B(G \cup \{a_x\}, n_{ai}; n_j)$, $target = n_j$.
    End if
  End for
  If $tag == true$, **then**
    $G = G \cup \{a_x\}$.
    For $\forall a_y \in IA(a_x)$ && $n_{ay} == n_{ai}$ && $f_y \neq 1$, **do**
      Insert Queue(Q, a_y) and set $f_y = 1$.
  End if
End while
If $G \neq \emptyset$, move G to the target node $target$.

IV. ANALYSES OF THE MODEL

A. CONVERGENCE ANALYSIS

In addition to evaluating the performance of the dynamic multiagent model on social execution cost, its convergence should also be judged. Motivated by the potential function concept that is often used to identify potential games [32], we have the following result.

Theorem 1: In Algorithm 1, each time that an agent team G moves from the current node to a preferable suitable node that achieves the bene t B, the social execution cost will reduce the value of B correspondingly.

Proof: Denote by $S=\{s_1,s_2;:::;s_n\}$ the agents' strategy pro le and $SEC(S)= SWC(S)=2+ SCC(S)/2$ the social execution cost function on S, where $SWC(S)=\sum_{n_i \in N} L_{n_i}(L_{n_i}+1)$, L_{ni} is the agent load on node n_i and $SCC(S)=\sum_{a_i;a_j \in A} d(s_i; s_j)$. Now, consider an agent team G that changes its strategy by moving from node s_G to node s'_G. From the perspective of the rst term $SWC(S)$ of $SEC(S)$, we have

$$\begin{aligned}
&SWC(s_G; S_{-G}) - SWC(s'_G; S_{-G}) \\
&= [L_{s_G}(L_{s_G}+1) + L_{s'_G}(L_{s'_G}+1) + \sum_{n_j \neq s_G; s'_G} L_{n_j}(L_{n_j}+1)] \\
&\quad -[(L_{s_G} - l_G)(L_{s'_G}+1-l_G) + (L_{s'_G}+l_G)(L_{s'_G}+1+l_G) \\
&\qquad + \sum_{n_j \neq s_G; s'_G} L_{n_j}(L_{n_j}+1)] \\
&= 2l_G(L_{s_G} - L_{s'_G} - l_G)
\end{aligned}$$

where l_G is the number of agents in G, and L_{sG}; $L_{s'G}$ are the agent load on s_G and s'_G at the strategy pro le S.

On the second term $SCC(S)$ of $SEC(S)$, we have:

$$\begin{aligned}
&SCC(s_G; S_{-G}) - SCC(s'_G; S_{-G}) \\
&= [2\sum_{a_i \in G}\sum_{a_j \in IA(a_i)} d(s_G; s_j) \\
&\quad + \sum_{a_j \in A \setminus G}\sum_{a_k \in IA(a_j) \setminus G} d(s_j; s_k)] \\
&\quad -[2\sum_{a_i \in G}\sum_{a_j \in IA(a_i)} d(s'_G; s_j) \\
&\quad + \sum_{a_j \in IA \setminus G}\sum_{a_k \in IA(a_j) \setminus G} d(s_j; s_k)] \\
&= 2\sum_{a_i \in G}\sum_{a_j \in IA(a_i)} (d(s_G; s_j) - d(s'_G; s_j))
\end{aligned}$$

On the other hand, from the perspective of the team G, we have

$$\begin{aligned}
&Ec(G; s_G; S_{-G}) - Ec(G; s'_G; S_{-G}) \\
&= [\sum_{1 \leq i \leq l_G} (L_{s_G} - l_G + i) \\
&\quad + \sum_{a_i \in G}\sum_{a_j \in IA(a_i)} d(s_G; s_j)] \\
&\quad -[\sum_{1 \leq i \leq l_G} (L_{s'_G} + l_G - l_G + i) \\
&\quad + \sum_{a_i \in G}\sum_{a_j \in IA(a_i)} d(s'_G; s_j)] \\
&= l_G(L_{s_G} - L_{s'_G} - l_G) \\
&\quad + \sum_{a_i \in G}\sum_{a_j \in IA(a_i)} (d(s_G; s_j) - d(s'_G; s_j))
\end{aligned}$$

Until this point, we can conclude that for every $s_G, s'_G \in N$:

$$\begin{aligned}
SEC(s_G; S_{-G}) - SEC(s'_G; S'_{-G}) = Ec(G; s_G; S_{-G}) \\
-Ec(G; s'_G; S`_{-G})
\end{aligned}$$

Therefore, we have Theorem 1. □

Based on Theorem 1, next we will show the convergence of the muliagent model and how fast it will converge to an equilibrium solution.

Theorem 2: Given a complex task allocation problem in social a network, where the number of network nodes is m; the diameter of the network is d; the number of subtasks is n; each subtask has k interdependent subtasks on average and the in uence coef cients and are integers. Algorithm 1 takes at most O(n^2+ nkd) steps to reach a stable equilibrium solution.

Proof: Recall the social execution cost de nition that is de ned in De nition 3: $SEC(\cdot)= SWC(\cdot) + SCC(\cdot)$. Note that at the initial state (i.e., the system agents are distributed on nodes randomly), we have $SWC(\cdot) \leq n(n+1)/2$ (the worst case with the maximum SWC value is that all of the agents queue at the same node and $SCC(\cdot) \leq nkd/2$ (the worst case with the maximum SCC value is that each pair of interdependent agents takes d hop distances to communicate. Note also that at the optimal state, $SEC(\cdot) \geq n(n/m+1)/2$ (the optimal case with the minimum SEC value is that agents are distributed on nodes evenly and interdependent agents queue at the same node without any communication cost). From Theorem 1, we know that each time a team changes its strategy, $SEC(\cdot)$ reduces at least

one unit value because and are integers. Thus, we can determine that $SEC(\cdot)$ will reach its minimum in at most $O(\ (n^2+n)+\ nkd-\ (n^2/m+n))=O(\ n^2+\ nkd)$ time steps. □

B. PERFORMANCE GUARANTEE ANALYSIS

Although the dynamic multiagent model can always converge to an equilibrium solution in polynomial time steps, its equilibrium solution is not necessarily the optimal solution that has the minimum social execution cost, even the agents are cooperative. Therefore, it is interesting and very much needed to analyze the multiagent model's degradation on system performance. The *price of anarchy* (*PoA*) measure (which is often used in game theory [33]) provides a good indicator to quantify the gap between the worst equilibrium solution and the optimal solution. In this paper, we use this notion to evaluate the multiagent model's performance. Let ES be the set of equilibrium solutions of the multiagent model; then, the price of anarchy of this model, *PoA*, can be de ned by the worst case ratio among all of the equilibrium solutions over the optimal solution (*Opt*) in terms of the social execution cost, i.e.,

$$PoA = \max_{S\in ES} SEC(S) \Big/ SEC(Opt) \tag{5}$$

Next, we will provide an upper bound of *PoA* of the multiagent model with the non-cooperative agents. As we discussed above, the cooperative setting might have a better solution than the non-cooperative setting, which will lead to a lower *PoA*. Thus, the multiagent model is likely to have a tight lower bound.

Theorem 3: Given a complex task allocation problem in a social network, where the number of network nodes is m; the diameter of the network is d; the number of subtasks is n; each subtask has k interdependent subtasks on average. The PoA of the multiagent model then is $O(1+3m(\ +2kd\ )/\ (m+n))$.

Proof: Let $S=\{s_1; s_2; \cdots; s_n\}$ and $P=\{p_1,p_2; \cdots; p_n\}$ be the equilibrium solution and the optimal solution, respectively. At strategy pro le S, the execution cost of the agent a_i is $Ec(s_i; S_{-i})= L_{s_i}(S)+ \sum_{a_j\in IA(a_i)} d(s_i; s_j)$, where $L_{si}(S)$ is the agent load on node s_i at strategy pro le S. The sum execution cost of all of the agents at S is:

$$\begin{aligned} Sum(S) &= \sum_{a_i\in A} Ec(s_i; S_{-i}) \\ &= \sum_{n_i\in N} L_{n_i}^2(S) + \sum_{a_i\in A}\sum_{a_j\in IA(a_i)} d(s_i; s_j) \\ &= 2SEC(S) - n \end{aligned}$$

As is known, at the equilibrium solution S, the execution cost of agent a_i should not decrease when a_i changes its strategy from s_i to p_i, i.e.,

$$\begin{aligned} & Ec(a_i; s_i; S_{-i}) \\ & \le Ec(a_i; p_i; S_{-i}) \\ &= (L_{p_i}(S)+1) + \sum_{a_j\in IA(a_i)} d(p_i; s_j) \\ &\le (L_{p_i}(S)+1) + \sum_{a_j\in IA(a_i)} (d(s_i; s_j) + d(s_i; p_i)) \end{aligned} \tag{6}$$

The inequality (6) follows from the triangle inequality for the social distance, i.e., $\forall n_i,n_j; n_k\in N$, $d(n_i; n_j)\le d(n_i; n_k) + d(n_k; n_j)$. If we sum the execution cost of all agents, we then have the bound of the social execution cost of the equilibrium solution $S;$ as follows:

$$\begin{aligned} Sum(S) &= \sum_{a_i\in A} Ec(a_i; s_i; S_{-i}) \le \sum_{a_i\in A} Ec(a_i; p_i; S_{-i}) \\ &\le \sum_{n_i\in N} L_{n_i}(P)(L_{n_i}(S)+1) \\ &\quad + (\sum_{a_i\in A}\sum_{a_j\in IA(a_i)} d(s_i; s_j) + 2nkd) \\ &= \sum_{n_i\in N} L_{n_i}(P)\cdot L_{n_i}(S) \\ &\quad + \sum_{a_i\in A}\sum_{a_j\in IA(a_i)} d(s_i; s_j) + n\ + 2nkd \\ &\le \sum_{n_i\in N} \frac{1}{2}(L_{n_i}^2(P) + L_{n_i}^2(S)) \\ &\quad + \sum_{a_i\in A}\sum_{a_j\in IA(a_i)} d(s_i; s_j) + n\ + 2nkd \end{aligned} \tag{7}$$

The inequality (7) follows from the fact $\forall x,\ y\in R$, $x^2+y^2\ge 2xy$. Derive from (7), we can conclude that

$$\begin{aligned} Sum(S) &\le \frac{1}{2}(\sum_{n_i\in N} L_{n_i}^2(S) \\ &\quad + \sum_{a_i\in A}\sum_{a_j\in IA(a_i)} d(s_i; s_j)) \\ &\quad + \frac{1}{2}\sum_{a_i\in A}\sum_{a_j\in IA(a_i)} d(s_i; s_j) \\ &\quad + \frac{1}{2}\sum_{n_i\in N} L_{n_i}^2(P) + n\ + 2nkd \\ &= \frac{1}{2}Sum(S) + \frac{1}{2}\sum_{n_i\in N} L_{n_i}^2(P) + n\ + 3nkd \\ &\Rightarrow Sum(S) \le \sum_{n_i\in N} L_{n_i}^2(P) + 2n\ + 6nkd \end{aligned}$$

By replacing $Sum\ (S)$ with $2SEC(S)$-n , we can derive

$$\begin{aligned} & 2SEC(S) - n \\ &\le \sum_{n_i\in N} L_{n_i}^2(P) + 2n\ + 6nkd \\ &\Rightarrow SEC(S) \le \sum_{n_i\in N} L_{n_i}^2(P) + \frac{3}{2}n\ + 3nkd \\ &\Rightarrow \frac{SEC(S)}{SEC(P)} \le \frac{\sum_{n_i\in N} L_{n_i}^2(P) + \frac{3}{2}n\ + 3nkd}{\sum_{n_i\in N} L_{n_i}^2(P)} \\ &\Rightarrow \frac{SEC(S)}{SEC(P)} \le 1 + \frac{3m(\ +2kd\ )}{(m+n)} \end{aligned} \tag{8}$$

The inequality (8) follows from

$$SEC(P) \ge \sum_{n_i\in N} L_{n_i}^2(P) \ge \frac{n(n+m)}{2m} \tag{9}$$

The inequality (9) has been proven in Theorem 2. Therefore, we have proven Theorem 3. □

V. EXPERIMENTAL VALIDATION AND ANALYSES

A. EFFECTIVENESS

We test the effectiveness of the mulitagent-based social complex task allocation model in small-scale settings. In these

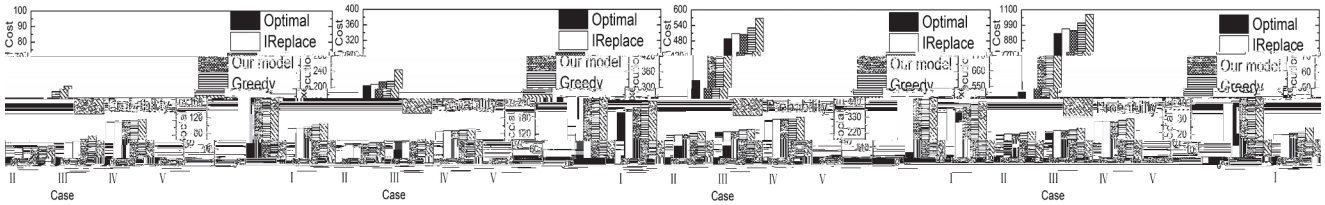

FIGURE 4. The performance of various task allocation models on different (α, β) values and different complex task sizes. (a) #Subtask=4. (b) #Subtask=8. (c) #Subtask=12. (d) #Subtask=16.

settings, each network consists of 30 nodes interconnected by a small-world structure in which the rewiring probability is p=0.2 [34]. The number of capabilities owned by each node $n_i \in N$ is given by $U(1,4)$ ($U(a; b)$ returns a value that is distributed at the interval $[a; b]$ uniformly), and each owned capability $c_j \in O_{ni}$ is chosen from the range [1, 16] randomly. There is only one complex task ct to be executed, and its subtask size is set to 4, 8, 12 or 16. Each subtask $t_i \in ct$ is randomly assigned a single required capability $R(t_i) \in$ [1, 16]. The underlying interdependent relationships of these subtasks are randomly generated: each subtask has a dependency with another subtask with a probability of p=0.4. Moreover, we select ve cases that have various coef cient values (i.e., and) for the calculation of the social execution cost, i.e., case I: (,)=(1; 5), case II: (,)=(1; 2) case III: (,)=(1; 1) case IV: (,)=(2; 1) and case V: (,)=(5; 1).

We conduct the multiagent model as follows: we rst allocate each subtask to the suitable node randomly and denote this solution as the initial solution. Then, we utilize our model to reallocate the subtasks by allowing the agents (who carry the subtasks) to change their node location. We compare our model (**Our model**) with the following four conventional task allocation models:

- **Centralized optimal model (Optimal)**, which utilizes an exponential brute-force search method to consider all of the possible allocations of subtasks to nodes.
- **Centralized iterative replace model (IReplace)** [2] where a central controller rst allocates each subtask to the most suitable node with the minimum task load and then replaces the allocated node of each subtask iteratively to reduce the social execution cost.
- **Centralized greedy model (Greedy)** [8], where the central controller identi es the best node for each complex task and allocates this complex task to that node. The best node means that it can allocate unsatis ed subtasks to its contextual nodes with the minimum social execution cost.
- **Distributed agent-based probability model (Probability)** [23], where each agent carries a task wanders from its current node to another node probabilistically. If an agent encounters a node that produces a smaller execution cost than the system's average value, it queues at that node; otherwise, it continues wandering.

Fig. 4 shows the nal social execution costs (*SEC*) produced by these models, which are achieved by averaging over 20 instances. From the experimental results, we conclude the following:

1) In all cases, our model performs very close to the Optimal and IReplace models on *SEC*, which is better than the Greedy and Probability models. This level of performance occurs because each agent team in our model strives to search for the optimal suitable node to queue at, which is bene cial to reduce the system's social execution cost.

2) In the latter two cases (i.e., cases IV and V, where the in uence of the waiting cost is larger than that of the communication cost), our model produces the less social execution cost compared to the rst two cases (i.e., cases I and II). The potential reason is that when waiting cost can overlap the communication cost, our model will take advantage of reducing the waiting cost over its generated inter-node communication cost. While in the rst two cases (where the in uence of the communication cost is larger than that of the waiting cost), the system expends much effort for communication, which implies that interdependent agents are more likely to queue at the same node because of the zero intra-node communication cost. As de ned in De nition 3, *SEC* linearly depends on agents' communication costs, while *SEC* is proportion to the square of each node's agent load (i.e., $SEC(\cdot) \sim O(L_{ni}^2)$. Thus, the larger the communication cost coef cients are, the larger the value of *SEC* that will be inlized50(will)-2oooo450[(thB.505(0(SC

tions,ork commupris co2000234(codes.)]T34(co)-49

nidels theracwork wat34rst atbri034n5-204(tos.)crib-

is

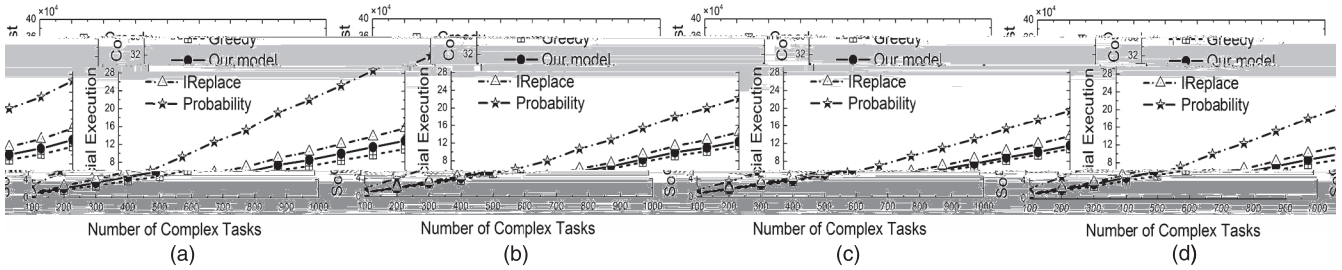

FIGURE 5. **The social execution costs of different models in the large-scale applications. (a) Small-World. (b) Scale-Free. (c) Scale-Free with TC. (d) Random.**

FIGURE 6. **The running times of different models in the large-scale applications. (a) Small-World. (b) Scale-Free. (c) Scale-Free with TC. (d) Random.**

we add a new node and connect this new node to m_1 nodes that already existing in the network. The probability that a new node v connects an existing vertex u is proportional to the degree of u [35].

- *Scale-Free Network With Triad Formation (Scale-Free With TF):* This network is built based on the scale-free network by adding an additional triad formation step: if a connection is added between nodes v and u, then another connection is added from v to a randomly selected neighbor of u [36]. This additional triad formation step constructs a network with power-law degree distribution and a high clustering coef cient.
- *Random Network:* By referring to [37], the random network is generated by randomly adding connections between agents with a probability of p=0.003, which results in the network average degree being equal to 6.

We assume that there are hundreds of complex tasks, whose numbers range from 100 to 1000, submitted to the system. The number of subtasks of each complex task is given by $U(4, 16)$. Due to space limitations, here we consider only the coef cient case (,)=(1; 1) (we also evaluate the system performance on other coef cient cases with different (,) values, and we obtain similar observations; thus, we omit the discussion of these cases). The other settings are similar to those described in Section 5.1.

Because **Optimal** is intractable in the large-scale settings, in this experiment, we compare only our model with the other three models, i.e., **Greedy**, **IReplace** and **Probability**. Fig. 5 shows the results of the *SEC*s of these models, from which we can conclude that: 1) In all of the experiments, our model performs slightly worse than Greedy but is much better than Probability and IReplace on *SEC*, which is especially notable when compared to the Probability. 2) In contrast to what we have observed in Section 5.1 that IReplace performs better than Greedy in small-scale applications, in large-scale applications, IReplace performs worse than Greedy. The potential reason is that, in the large-scale applications, there are thousands of subtasks, and at each iterative round, IReplace considers only the current chosen subtask, ignoring the status of its direct (or indirect) relevant subtasks (the numbers of these subtasks are considerable in the large-scale applications), while Greedy can alleviate this problem in these scenarios.

Fig. 6 shows the running times of these models, from which we can observe that: 1) compared to our model, the traditional Greedy, IReplace and Probability models spend much more time on task allocation. For example, in the case that #complex task=1000 in a small-world structure (i.e., Fig. 6(a)), Greedy must spend one hour and a half (approximately 5×10^3(s)) to return the allocation result, while our model requires only several minutes. We explain this phenomenon by analyzing these models' theoretical computational complexity. Given a social task allocation problem that has m nodes,n complex tasks and each complex task consists of k subtasks on average, the computation complexity of Greedy and IReplace is $O(nk^2m^2)$ (The details of the complexity description of the IReplace and Greedy models can be found in [2] and [8]). As discussed in Section 5.1, our model takes at most $O(k^2n^2)$ time steps to converge to a stable equilibrium, and at each time step, an agent needs to take only $O(k)$ operations to compute the most bene cial team. Thus, the time complexity ratio

between Greedy (or IReplace) and our model then equals to $O(nk^2m^2)/O(k^3n^2)=m^2/(kn)$, which is consistent with the experimental results to some extent.

Table 1 shows the properties (e.g., network diameter, characteristic path length (*CPL*) and clustering coef cient) of these networks used in this experiment. Fig. 7 shows the *SEC* produced by our model in these networks. From the results shown in Fig. 7, it can be found that our model is more relevant to network diameter and *CPL*: if the network has a shorter diameter and a smaller *CPL*, our model will produce the less *SEC*. For example, our model produces the least *SEC* in the Scale-Free with *TF* network that has the shortest diameter and smallest *CPL* compared to other networks (i.e., Small-World, Scale-Free and Random). On the other hand, the clustering coef cient feature does not show direct correlation with the performance of our model. For example, although Small-World has the higher clustering coef cient than that in the scale-free with *TC*, our model produces the less*SEC* in Scale-Free with *TC* than the *SEC* in Small-world.

TABLE 1. The properties of networks.

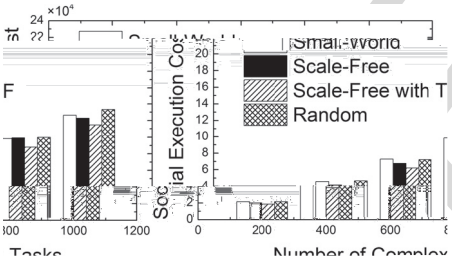

FIGURE 7. The effect of the network properties on social execution cost.

To summarize, our model is a desirable option for the large-scale applications where quality performance and real-time response are highly required.

C. ROBUSTNESS

Social networks are inherently open and dynamic with user turnover and connection changes [38], [41]. An ef - cient social task allocation model should also be robust to be capable of addressing the network dynamics. We test the robustness of our model on the networks' topology dynamics: initially, we utilize our model to allocate two complex tasks (each consists of 16 subtasks) on a small-world network (which is composed of 30 nodes that are interconnected by a small-world structure). During the task allocation process, four disturbance events occur sequentially, which are described as follows:

- *Event 1 (Connection Break):* Because of the interest con- ict, a set of connected individuals break their connections (here we set each pair of pairwise connected nodes with the probability p=0.2 of breaking their connection).
- *Event 2 (Connection Enhance):* Because of the occasional cooperation experience, some strange individuals become friends (here, we set each pair of pairwise unconnected nodes with the probability p=0.2 of being interconnected).
- *Event 3 (Individuals Entrance):* Because of the recommendation of the existing users, some new individuals register and enter into this network (here, we assume that there are 5 new nodes that arrive to the system).
- *Event 4 (Individuals Exit):* Because it is time consuming to sustain the memberships with their connections, some individuals can log out and leave this network (here, we assume that there are 5 nodes that exit the system).

The rst two events change the connections of the network, and they are designed to test our model's ability to adjust the social communication cost. The remaining two events change the number of nodes, and they are designed to test our model's ability to adjust the social waiting cost. The resulting social execution cost plot with the above four disturbance events is shown in Fig. 8, from which we observe that once the disturbance occurs, the *SEC* changes immediately. However, our model can adapt to the disturbance within several seconds and can converge quickly to another stable desirable solution. It should also be noted that in Fig. 8, as our model proceeds, the *SEC* decreases as well, making our model an anytime model: the task allocation process can be terminated at any time, where it can provide the system with a solution that is better than any of the preceding states. Moreover, our model can always converge to stable equilibrium in nite time steps.

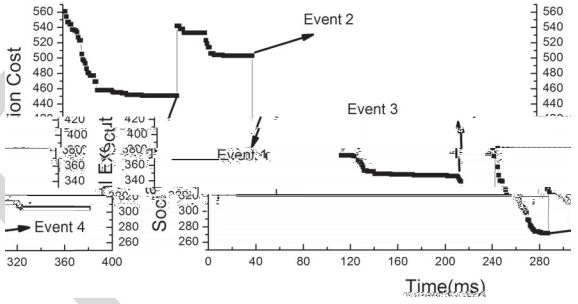

FIGURE 8. The social execution cost plot with four disturbance events.

VI. RELATED WORK

In this section, we rst review the traditional task allocation researches in social networks and then provide a brief discussion of the multiagent-based task allocation technology that has been applied to other networked systems.

A. TASK ALLOCATION IN SOCIAL NETWORKS

Given a task T and social network SN consisting of various individuals, one of the main objectives of social task allocation is to allocate T to a set of professional individuals $I \subseteq SN$ in such a way that I can collaborate effectively [2], [3], [8] [10]. Lappas et al. [2] refer to social task allocation as *social team formation* and attempt to build an ef cient team such that the team not only satis es the capability requirements of a task but also has the smallest team cost. This letter of Lappas et al. [2] was further investigated by several team formation variants with additional goals and constraints [3], [8] [10]. For example, Kargar and An [8] assume the existence of a team leader, and the team cost is measured by the summed distance between the team leader and the team members. Datta et al. [3] and Anagnostopoulos et al. [9] believe that team formation in social networks should not only optimize the social effectiveness but also address the load balancing (i.e., the workloads allocated to each expert should be proportional to his capacity). Rangapuram et al. [10] investigate a more realistic social team formation problem by introducing more generalized constraints, such as *i*) including a predetermined team leader; *ii)* the team members should be socially close and *iii*) the bounded team budget. Note that all of these problems are NP-hard and the previous researchers mainly focus on developing centralized approximations that have a high performance guarantee. However, the low robustness and high computational complexity prevent the centralization from scaling well to large-scale systems in which there are millions of social individuals to consider and thousands of tasks to be executed [19], [20].

B. TASK ALLOCATION IN NETWORKED MULTIAGENT SYSTEMS

In this type of study, each individual is modeled as a sel sh agent whose aim is to maximize its own pro t. Market-based mechanisms can be well exploited by the agents to perform tasks [13], [19], [26], [27]. For example, in an agent network, to optimize an agent's own bene t, the agent can make a contract with its neighbors about which task to undertake [13], [19]. When the agents have incomplete information on other agents' resource prices, they can utilize a bilateral bargaining protocol to negotiate with others round-by-round until they have made an agreement on the resource price [26], [27]. On the other hand, the network structure itself can affect system performance on task completion [25], [28], [29]. Gaston and desJardins [28] and Kota et al. [25] thus develop a structural adaptation method to increase social welfare, where agents can adjust the network structure by deleting their costly connections and rewiring them to those agents that have better connections.

Besides system monetary revenue, the task resource access time in the networked system is also a crucial factor of the system performance [6], [12], [14], [15]. To reduce the system resource access time, Jiang and Jiang [6] present a contextual resource negotiation mechanism by allowing agents to negotiate with others from nearby to faraway gradually. To achieve dependable resources with the least resource access time for undependable social networks, Jiang et al. [12] propose a reputation-based negotiation mechanism. Recently, a network-layer oriented task allocation model is presented for minimizing the task execution time in multiplex networks [14]. By being aware of the community structure in networks, Wang and Jiang [15] propose a community-aware task allocation model (where agents can cooperate with other agents in the same community) to improve social welfare while incurring a few of negotiation overhead. In the distributed network computing systems (e.g., grids), the nodes (e.g., computers, machines or workstations) have to take time to execute the tasks, and thus, the primary goal in this kind of system is to maximize throughput. To complete the tasks as soon as possible, Liu et al. [23] propose an agent-based probability load balancing method to distribute the tasks on nodes evenly.

All of these above research approaches are ef cient for the independent task allocation problems in which there are no dependencies among the tasks. While this paper focuses on addressing the interdependent task allocation in social networks, where the success of a task also depends on how effectively the involved individuals communicate. In reality, in case two experts have negative relationships, they are unlikely to complete the interdependent tasks successfully even if they are professional in these activities [18].

More broadly, this social task allocation problem can also be viewed as a speci c variant of the constraint satisfaction problem (CSP), and hence, some related distributed CSP optimizations such as ADOPT [39] and cooperative mediation-based systems [40]. However, because of the multi-stage inter-node negotiations, these methods will produce prohibitive network traf c overhead, which is unacceptable for practical online applications [24]. Compared to these studies, we restrict agents to cooperate only with their intra-node agents that queue at the same node.

VII. CONCLUSIONS AND FUTURE WORK

In this paper, we address the complex task allocation in social networks, where a set of individuals should work together to satisfy a complex task's skill requirements. Moreover, this social task allocation should not only meet the traditional objective of load balancing, but also the new objective of maximizing social effectiveness. To meet both of the two objectives, we propose a distributed multiagent-based task allocation model by dispatching a mobile and cooperative agent to each subtask to search for the suitable individual that has the necessary skills, small workloads and lower coordination costs with others. Our experimental results show that our model produces as less task execution cost as the benchmark centralized models but reduces the computation time signi cantly compared to the traditional models. Moreover, our model adapts to network dynamics quickly, making it scale well in dynamic large-scale applications.

There are two interesting issues that can be investigated further. In this study, each agent (or agent team) searches in all the network nodes and chooses the one with the lowest execution cost as the target node. This global view of the environment might be unpractical in some real-world applications. In the future, we would like to devise more ef cient cooperation mechanisms (e.g., agents exchange their node location) to improve the performance of the system with local view constraint. Another limitation of this study is that the tradeoff coef cients between the waiting cost and communication cost are set to be xed. In reality, due to system dynamics such as frequent users and tasks turnover, xed coef cients cannot always optimize system performance (even might have a negative impact). Therefore, in the future, we would like to devise automatic adaption mechanism to dynamically adjust the coef cients to the change of environments rather than set them to xed values.

REFERENCES

[1] 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.

[2] T. Lappas, K. Liu, and E. Terzi, ``Finding a team of experts in social networks,'' in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, Paris, France, Jun./Jul. 2009, pp. 467 475.

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

[4] L. Tran-Thanh, S. Stein, A. Rogers, and N. R. Jennings, ``Ef cient crowdsourcing of unknown experts using multi-armed bandits,'' in *Proc. 20th Eur. Conf. Artif. Intell. (ECAI)*, Montpellier, France, Aug. 2012, pp. 768 773.

[5] 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. Auto. Agents Multiagent Syst. (AAMAS)*, Paris, France, May 2014, pp. 261 268.

[6] Y. Jiang and J. Jiang, ``Contextual resource negotiation-based task allo-

[39] P. J. Modi, W.-M. Shen, M. Tambe, and M. Yokoo, ``Adopt: Asynchronous distributed constraint optimization with quality guarantees,'' *Artif. Intell.*, vol. 161, nos. 1 2, pp. 149 180, Jan. 2005.
[40] R. Mailler and V. Lesser, ``A cooperative mediation-based protocol for dynamic distributed resource allocation,'' *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 36, no. 1, pp. 80 91, Jan. 2006.
[41] 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.

WANYUAN WANG (S'13) received the B.S. degree in information and computing science from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, 2011. He is currently pursuing the Ph.D. degree with the Distributed Intelligence and Social Computing Laboratory, School of Computer Science and Engineering, Southeast University. His main research interests include social networks and multiagent systems. He received the Best Student Paper Award from the IEEE International Conference on Tools with Arti cial Intelligence (ICTAI) in 2014. He has authored several articles in refereed journals and conference proceedings, such as the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, the IEEE TRANSACTIONS ON CYBERNETICS, and ICTAI.

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 and the Director of the Distributed Intelligence and Social Computing Laboratory with the School of Computer Science and Engineering, Southeast University, Nanjing, China. His main research interests include multi-agent systems, social networks, social computing, and complex distributed systems. He is a Senior Member of the China Computer Federation and the Chinese Institute of Electronics. He received the best paper award and the Best Student Paper Award from PRIMA and ICTAI, respectively. He has authored over 80 scienti c articles in refereed journals and conference proceedings, such as the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, the IEEE TRANSACTIONS ON CYBERNETICS, the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS, the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS PART C: APPLICATIONS AND REVIEWS, the *ACM Transactions on Autonomous and Adaptive Systems*, the *Journal of Parallel and Distributed Computing*, the International Joint Conference on Arti cial Intelligence, and the International Conference on Autonomous Agents and Multiagent Systems. He is also a member of the Editorial Board of *Advances in Internet of Things* and the *Chinese Journal of Computers*, and an Editor of the *International Journal of Networked Computing and Advanced Information Management* and *Operations Research and Fuzziology*.