Multiagent-Based Allocation of Complex Tasks in Social Networks

Wanyuan Wang, *Student Member*, *IEEE*, Yichuan Jiang*, *Senior Member*, *IEEE*

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 study, 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 beneficial. 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

1 INTRODUCTION

oday'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 nication [12-15].

By only considering building a team of engineers that can satisfy all of the skill requirements of *P*, the manager can allocate activities {*RA*, *AD*, *IM*} to a_2 and {*TE*, *DE*, *MA*} to a_6 (, Fig. 1(c)). However, in Scheme 1, engineers a_2 and a_6 might be overburdened with multiple activities, and these activities would have to wait for a long time to be executed. By considering alleviating the heavy workloads of the employees, the manager can allocate the activities {*RA*}, {*AD*}, {*IM*}, {*TE*}, {*DE*} and {*MA*} to engineers a_i (i=1,…,6), respectively (, Fig. 1(d)). In Scheme 2, however, the engineers might spend too much effort on communication when conducting interdependent activities, which will also degrade the benefit of product *P*. By considering improving the employees' communication efficiency, the manager can allocate {*RA*} to a_1, {*AD*, *IM*} to a_2, {*MA*} to a_3 and {*TE*, *DE*} to a_6 (, Fig. 1(e)). Scheme 3 is superior to Scheme 2 in that it employs a team of well-connected engineers that can collaborate with the less communication cost while preserving their small workloads.

Therefore, an efficient social task allocation approach should not only focus on finding a set of employers that have the necessary skills and a fair allocation of the workloads among the employers but also address the objective of maximizing the social effectiveness. Social effectiveness is the indicator used to determine how effectively social individuals can communicate. In many SNs, social connection always represent a positive relationship among individuals, such as friendship in acquaintance networks [16], partnership in collaboration networks [5] and location proximity in opportunistic mobile networks [17]. Therefore, in this paper, we simply use the measure *social distance* to quantify the social effectiveness: socially closer individuals yield higher social effectiveness (i.e., they can communicate with one another more effectively) [18].

To the best of our knowledge, the first work to address the objective of maximizing the social effectiveness when addressing the social task allocation problem was by Lappas et al. [2]. The work of Lappas et al. [2] was furthered investigated by considering several variants with some additional constraints [3][8-10]. Unfortunately, all of these studies require a central authority that maintains information on all of the users and can communicate with them directly for task allocation in a centralized manner. Although centralization can provide a rigorous quality guarantee, the poor robustness and high computational complexity make it unacceptable for large-scale SNs [19][20]. Against this background, this paper introduces a novel distributed task allocation model by utilizing multiagent technology [21-24]. In the proposed model, immediately when a complex task arrives to the system, a mobile and cooperative agent is dispatched to each subtask of the complex task. With respect to the mobility, each agent can transport itself to a suitable user in such a way that the user *i*) can perform the subtask it carries, *ii*) can communicate with other users effectively and *iii*) undertakes small workloads. With respect to cooperativeness, agents can cooperate with each other by forming teams and moving to a suitable user jointly if cooperation can mitigate load unfairness and increase social effectiveness.

Thus, the main contribution of this paper is to propose a distributed social task allocation model for the first time with the aims of both load balancing and social effectiveness maximization. Moreover, because our cooperation mechanism is implemented by the agents that queue at the same user, it avoids causing heavy network traffic for agents' negotiations. We evaluate the proposed model both theoretically and experimentally. Through theoretical analyses and experiments, we determine that the proposed model 1) can converge to a stable solution in polynomial time; 2) is efficient in improving the system load balance and social effectiveness; 3) reduces the computation time significantly compared to all conventional models and 4) adapts to network dynamics 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 definition 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 effec-
[illegible]

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 reflect 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*.

Given a valid allocation Φ *of subtask* $\forall t_{ij}\in CT_i$ $(CT_i\in\Gamma)$ *to a node* $n_k\in N$*, the social waiting cost of all of the subtasks,* $SWC(\Phi)$*, is defined as follows:*

$$SWC\ \Phi\ = \sum_{n_i\in N} L_{n_i}\ L_{n_i} + \ / \tag{1}$$

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 first 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 $\sum_{\le i\le L_{n_i}} i = L_{n_i}\ L_{n_i} +$.

This social waiting cost definition is motivated by [23], which is powerful enough to quantify the load balance extent.

Given a social task allocation problem, the smaller the social waiting cost of a valid allocation Φ*, the more balanceable the allocation* Φ *is.*

2.2.2 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.

Given a valid allocation Φ *of subtask* $\forall t_{ij}\in CT_i$ $(CT_i\in\Gamma)$ *to a node* $n_k\in N$*, the social communication cost of all subtasks,* $SCC(\Phi)$*, is defined as follows:*

$$SCC\ \Phi\ = \sum_{CT_i\in\Gamma}\sum_{t_{ij}\in CT_i}\sum_{t_{ik}\in\Omega\ t_{ij}} d\ n\ t_{ij}\ \ n\ t_{ik}\ / \tag{2}$$

Given a social task allocation problem, the smaller the social communication cost of a valid allocation Φ*, the more socially effective the allocation* Φ *is.*

2.3 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 conflicting. 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).

with the mobility property. With respect to mobility, we mean that each agent a_i can transport itself to a wthe1 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 3.1, we first 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 3.2, we develop an efficient multiagent-based task allocation model with cooperative agents. tionproperty. 1

1 a itself

3.1 The Model with Non-Cooperative Agents

In the non-cooperati
v,coo5 44.3TD0.0016Tc(the)Tj/TT121Tf1.39240T30TD0Tc<00231>Tj/TT131Tf2.25320TD-0024Tc[(mod)5.5(e)-Tf1.39

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 traffic overhead produced by the intra-node negotiation is so small that it can be neglected [24].

3.2 The Model with Cooperative Agents

The main idea of the cooperation mechanism implemented by the agents can be briefly 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 defined as follows:

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:*

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

The first 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 first 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 $\sum_{\leq i \leq l_G} L_{n_G} - l_G + i$. The second term represents the total communication cost of G.

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 profile 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 definition, a_1 finds 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 $Ec\ G\ n\ = \alpha \sum_{i=} L_{n_G} - l_G + i\ + \beta \sum_{a_i \in G} \sum_{a_j \in IA\ a_i} d\ n_G\ n_{a_j}\ =\ \alpha =$, after the team G moves to n_1, the execution cost of G becomes $Ec(G,n_1)=6\alpha+\beta=23<27=Ec(G,n_2)$. 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 (

-

) 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 benefit to quantify how much an agent team gains by changing its strategy. The benefit 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 benefit definition, 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., $a_j \notin \bigcup_{a_i \in G} IA\ a_i$) 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.

For any agent $a_i \in A$, it is only beneficial 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 $\psi(a_i)$ can be further limited within its intra-node interdependent agents and interdependent agents' interdependent agents, if necessary, which can be denoted by $\psi(a_i)=\{a_j|n_{aj}=n_{ai} \wedge ST(a_j).CT=ST(a_i).CT\}$ ($ST(a_i).CT$ means the complex task that subtask $ST(a_i)$ belongs to). Nevertheless, for each agent, even finding the optimal team that yields the largest benefit 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 $\psi(a_i)$ must consider the exponential number $O(2^{|\psi(ai)|})$ of possible combinations. To deal with this computationally costly optimization problem, we propose an efficient *Breadth-First* negotiation mechanism, where each agent forms a beneficial 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).

. Suppose that the four interdependent agents {a_i|1≤i≤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 beneficial team then can be described as in Fig. 3(b): first, a_1 negotiates with its direct interdependent agents a_2 and a_3, i.e., the agents in gradation 1. If a_1 finds that it is beneficial 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 1

[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., $\forall a_i \in G$, $R(ST(a_i)) \in On_G$.

Multiagent-Based Social Complex Task Allocation Model

the following procedure until no agent team can benefit by changing its strategy.
an agent $a_i \in A$ randomly.
the flag f_j of agent a_j queuing at n_{a_i} to 0.
G=Ø, *max*=0 and *target*=Null.
Queue(Q).
Queue(Q, a_i), and set f_i=1.
(Q≠Ø)
a_x= Queue(Q) and tag=*false*.
$\forall n_j \in N$
$B(G \cup \{a_x\}, n_{a_i}, n_j)$ *max* &&
$\forall a_y \in G \cup \{a_x\}$, $R(ST(a_y)) \in O_{n_j}$
tag=*true*, *max*=$B(G \cup \{a_x\}, n_{a_i}, n_j)$, *target*=$n_j$.
tag==*true*,
$G=G \cup \{a_x\}$.
$\forall a_y \in IA(a_x)$&& n_{a_y}==n_{a_i} && $f_y \neq 1$,
Queue(Q, a_y) and set f_y=1.
G≠Ø, move G to the target node *target*.

most beneficial team within its cooperation domain agents $\psi(a_i)$ by the *Breadth-First* negotiation mechanism takes only $O(|\psi(a_i)|)$ operations. □

We now give a formal description of the multiagent-based social complex task allocation model (see Algorithm 1). In Algorithm 1, for each randomly chosen agent a_i, it first initializes its state before strategy changing (steps 2~4). The set G stores the agent team that a_i would like to move jointly, the variable *target* indicates the destination node that G prefers to move to, and the value *max* records the current maximal benefit that the team G gains by changing strategy. Agent a_i then utilizes the *Breadth-First* negotiation mechanism to form the most beneficial team with the agents in its cooperation domain (steps 5~19). If moving together with an agent a_x to the suitable node n_j produces less execution cost for the current team $G \cup \{a_x\}$, agent a_i will negotiate with a_x to join the team G (steps 10-12). Finally, if a_i realizes that it is beneficial to move from the current node to the destination node *target* by the team G (i.e., G≠Ø), team G will change its strategy by moving to the node *target* (step 20). The system terminates if no agent or agent team has any incentive to change strategy (step 1), and the final stable solution can be called an equilibrium solution.

4 Analyses of the Model

4.1 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.

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

Denote by $S=\{s_1,s_2,\ldots,s_n\}$ the agents' strategy profile and $SEC(S)=\alpha SWC(S)/2+\beta SCC(S)/2$ the social execution cost function on S, where $SWC\ S = \sum_{n_i \in N} L_{n_i}\ L_{n_i} +$, L_{n_i} 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 first term $SWC(S)$ of $SEC(S)$, we have

$$SWC\ \ s_G\ \ S_{-G}\ \ - SWC\ \ s_G\ \ S_{-G}$$
$$= L_{s_G}\ \ L_{s_G} + \quad + L_{s_G}\ \ L_{s_G} + \quad + \sum_{n_j \neq s_G\ s_G} L_{n_j}\ \ L_{n_j} +$$
$$- \quad L_{s_G} - l_G\ \ L_{s_G} + \ - l_G\ \ + \ L_{s_G} + l_G\ \ L_{s_G} + \ + l_G$$
$$+ \sum_{n_j \neq s_G\ s_G} L_{n_j}\ \ L_{n_j} +$$
$$= \ l_G\ \ L_{s_G} - L_{s_G} - l_G$$

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

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

$$SCC\ \ s_G\ \ S_{-G}\ \ - SCC\ \ s_G\ \ S_{-G}$$
$$= \ \sum_{a_i \in G} \sum_{a_j \in IA\ \ a_i} d\ \ s_G\ \ s_j\ \ + \sum_{a_j \in A\ \ G} \sum_{a_k \in IA\ \ a_j\ \ G} d\ \ s_j\ \ s_k$$
$$- \ \sum_{a_i \in G} \sum_{a_j \in IA\ \ a_i} d\ \ s_G\ \ s_j\ \ + \sum_{a_j \in IA\ \ G} \sum_{a_k \in IA\ \ a_j\ \ G} d\ \ s_j\ \ s_k$$
$$= \sum_{a_i \in G} \sum_{a_j \in IA\ \ a_i} \ d\ \ s_G\ \ s_j\ \ - d\ \ s_G\ \ s_j$$

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

$$Ec\ \ G\ \ s_G\ \ S_{-G}\ \ - Ec\ \ G\ \ s_G\ \ S_{-G}$$
$$= \ \alpha \sum_{\leq i \leq l_G} \ L_{s_G} - l_G + i\ \ + \beta \sum_{a_i \in G} \sum_{a_j \in IA\ \ a_i} \ d\ \ s_G\ \ s_j$$
$$- \ \alpha \sum_{\leq i \leq l_G} \ L_{s_G} + l_G - l_G + i\ \ + \beta \sum_{a_i \in G} \sum_{a_j \in IA\ \ a_i} \ d\ \ s_G\ \ s_j$$
$$= \alpha l_G\ \ L_{s_G} - L_{s_G} - l_G\ \ + \beta \sum_{a_i \in G} \sum_{a_j \in IA\ \ a_i} \ d\ \ s_G\ \ s_j\ \ - d\ \ s_G\ \ s_j$$

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

$$SEC(s_G,S_{-G})\text{-}SEC(s'_G,S'_{-G})=Ec(G,s_G,S_{-G})\text{-}Ec(G,s'_G,S'_{-G})$$

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.

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 influence coefficients α and β are integers. Algorithm 1 takes at most $O(\alpha n^2+\beta nkd)$ *steps to reach a stable equilibrium solution.*

Recall the social execution cost definition that is defined in Definition 3: $SEC(\cdot)=\alpha SWC(\cdot)+\beta SCC(\cdot)$. 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, $SEC(\cdot) \geq \alpha n(n/m+1)/2$ (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 α and β are integers. Thus, we can determine that $SEC(\cdot)$ will reach its minimum in at most $O(\alpha(n^2+n)+\beta nkd-\alpha(n^2/m+n))= O(\alpha n^2+\beta nkd)$ time steps. □

4.2 Performance Guarantee Analysis

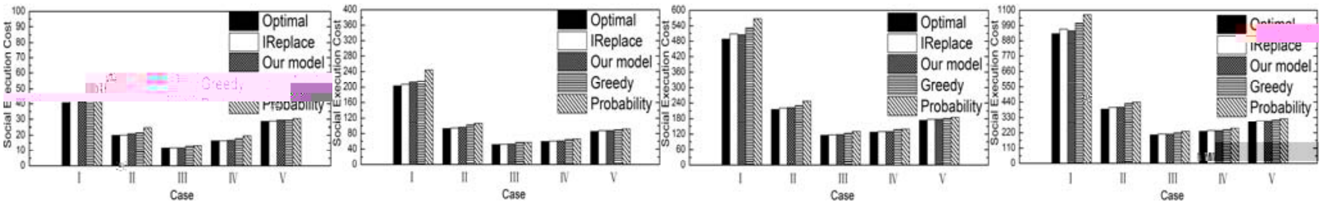
Although the dynamic multiagent model can always converge to an equilibrium solution in polynomial time steps, its equilibrium solution is not necessarily the optimal solution that has the minimum social execution cost, even the agents are cooperative. Therefore, it is interesting and very much needed 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 defined 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.,

$$PoA = \underset{S\in ES}{} SEC\ S \Big/ SEC\ Opt \tag{5}$$

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.

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 $O(1+3m(\alpha+2kd\beta)/\alpha(m+n))$.

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



(a) #Subtask=4 (b) #Subtask=8 (c) #Subtask=12 (d) #Subtask=16

Fig. 4. The performance of various task allocation models on different (α β values and different complex task sizes.

- , which utilizes an exponential brute-force search method to consider all of the possible allocations of subtasks to nodes.
- [2] where a central controller first 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.
- [8], where the central controller identifies the best node for each complex task and allocates this complex task to that node. The best node means that it can allocate unsatisfied subtasks to its contextual nodes with the minimum social execution cost.
- - - [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 final 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 beneficial to reduce the system's social execution cost.

2) In the latter two cases (i.e., cases and , where the influence of the waiting cost is larger than that of the communication cost), our model produces the less social execution cost compared to the first two cases (i.e., cases and). 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 first two cases (where the influence 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 defined in Definition 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 coefficients are, the larger the value of *SEC* that will be incurred.

5.2 Scalability

We test the scalability of our model in large-scale settings, where each network comprises 2000 nodes. In this experiment, four typical network models (i.e., Small-World, Scale-Free, Scale-Free with Triad Formation and Random network) are used to imitate the underlying topology of these nodes. Before discussing the results of the task allocation models on these networks, we first briefly describe how these networks are constructed.

- *Small-World Network.* This network starts from a regular ring lattice in which each node connects with 6 nearest neighbors, and this node has a probability of p=0.2 of rewiring each connection to another node [34].
- *Scale-Free Network.* The scale-free network starts with m_0 nodes connected by m_0-1 connections. At each step, 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 coefficient.
- *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 coefficient case (α, β)=(1,1) (we also evaluate the system performance on other coefficient cases with different (α β 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 is intractable in the large-scale settings, in this experiment, we compare only our model with the other three models, i.e., , and . 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][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 beneficial 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 I shows the properties (e.g., network diameter, characteristic path length (*CPL*) and clustering coefficient) 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 coefficient feature does not show direct correlation with the performance of our model. For example, although Small-World has the higher clustering coefficient 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.

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

5.3 Robustness

Social networks are inherently open and dynamic with user turnover and connection changes [38][41]. An efficient 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-

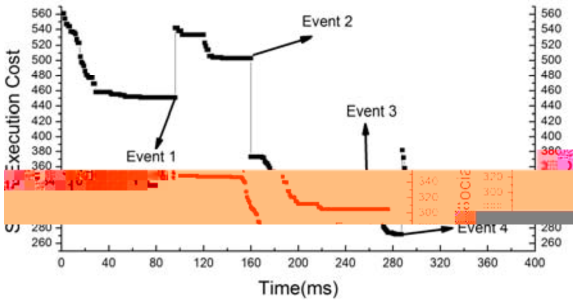


Fig. 8. The social execution cost plot with four disturbance events.

- 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 first 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 finite time steps.

In this section, we first 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.

6.1 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 efficient team such that the team not only satisfies the capability requirements of a task but also has the smallest team cost. The work 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].

6.2 Task Allocation in Networked Multiagent Systems

In this type of study, each individual is modeled as a selfish agent whose aim is to maximize its own profit. 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 benefit, 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 efficient 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 specific 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 traffic 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.

7 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 significantly 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 efficient 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 coefficients between the waiting cost and communication cost are set to be fixed. In reality, due to system dynamics such as frequent users and tasks turnover, fixed coefficients 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 coefficients to the change of environments rather than set them to fixed values.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (No.61170164, and No. 61472079), the Funds for Distinguished Young Scholars of the Natural Science Foundation of Jiangsu Province (No.BK2012020), and the Program for Distinguished Talents of Six Domains in Jiangsu Province (No.2011-DZ023).

REFERENCES

[1] Yichuan Jiang and J.C. Jiang, "Understanding Social Networks from a Multiagent Perspective," *IEEE Transactions on Parallel and Distributed Systems*, 25(10):2743-11 1

mental Study," *Games and Economic Behavior*, 66(1): 506-525, 2009.

[19] Mathijs de Weerdt, Yingqian Zhang and Tomas Klos, "Multiagent Task Allocation in Social Networks," *Autonomous Agents and Multi-Agent Systems*, 25(1): 46-86, 2012.

[20] Thuan Duong-Ba, Thinh Nguyen, Duc A. Tran, "DAROS: Distributed User-Server Assignment and Replication for Online Social Networking Applications," *Proceedings of the 2013 International Conference on Computing, Networking and Communications (ICNC-13)*, pp.456-460, San Diego, USA, January 28-31, 2013.

[21] Federico Bergenti, Enrico Franchi and Agostino Poggi, "Agent-based Social Networks for Enterprise Collaboration," *Proceedings* of the *IEEE 20th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'11)*, pp.25-28, Paris, France, June 27-29, 2011.

[22] Ka-Po Chow and Yu-Kwong Kwok, "On Load Balancing for Distributed Multiagent Computing," *IEEE Transactions on Parallel and Distributed Systems*, 13(8): 787-801, 2002.

[23] Jiming Liu, Xiaolong Jin and Yuanshi Wang, "Agent-based Load Balancing on Homogeneous Minigrids: Macroscopic Modeling and Characterization," *IEEE Transactions on Parallel and Distributed Systems*, 16(7): 586-598, 2005.

[24] Tino Schlegel, Peter Braun and Ryszard Kowalczyk, "Towards Autonomous Mobile Agents with Emergent Migration Behaviour," *Proceedings of the 5th International Conference on Autonomous Agents and Multiagent systems (AAMAS'06)*, pp.585-592, Hakodate, Japan, May 8-12, 2006.

[25] Ramachandra Kota, Nicholas Gibbins and Nicholas R. Jennings, "Self-Organising Agent Organisations," *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'10)*, pp.797-804, Budapest, Hungary, May 10-15, 2009.

[26] Bo An, Victor Lesser, David Irwin and Michazel Zink, "Automated Negotiation with Decommitment for Dynamic Resource Allocation in Cloud Computing," *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'10)*, pp.981-988, Toronto, Canada, May 10-14, 2010.

[27] Dayong Ye, Minjie Zhang and Danny Sutanto, "Self-Adaptation-Based Dynamic Team Formation in A Distributed Agent Network: A Mechanism and A Brief Survey," *IEEE Transactions on Parallel and Distributed Systems*. 24(5): 1042-1051, 2013.

[28] Matthew E. Gaston and Marie desJardins, "Agent-Organized Networks for Dynamic Team Formation," *Proceedings of the 4th international Conference on Autonomous Agents and Multiagent Systems* (*AAMAS'05*), pp.230-237, Utrecht, Netherlands, July 25-29, 2005.

[29] Bijan Ranjbar-Sabraei, Haitham Bon Ammar, Daan Bloembergen, Karl Tuyls and Ggerhard Weiss, "Evolution of Cooperation in Arbitrary Complex Networks," *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'2014)*, pp.667-684, Paris, France May 5-9, 2014.

[30] Jonathan P. Pearce, Milind Tambe and Rajiv Maheswaran, "Solving Multiagent Networks Using Distributed Constraint Optimization," *AI Magazine*, 29(3): 47-62, 2008.

[31] Xiaoming Zheng and Sven Koenig, "K-Swaps: Cooperative Negotiation for Solving Task-Allocation Problems," *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI'09)*, pp. 373-378, Pasadena, California, USA, July 11-17, 2009.

[32] Dov Monderer and Lloyd S. Shapley, "Potential Games," *Games and Economic Behavior*, 14(1): 124-143, 1996.

[33] Geogre Christodoulou and Elias Koutsoupias, "The Price of Anarchy of Finite Congestion Games," *Proceedings of the 37th annual ACM symposium on Theory of computing (STOC'05)*, pp.67-73, Baltimore, USA, May 22-24, 2005.

[34] Duncan J. Watts and Steven H. Strogatz, "Collective Dynamics of 'Small-World' Networks," *Nature*, 393(6684): 440-442, 1998.

[35] Albert László Barabási and Réka Albert, "Emergence of Scaling in Random Networks," *Science*, 286(5439): 509-512, 1999.

[36] Federico Bergenti, Enrico Franchi, and Agostino Poggi, "Selected Models for Agent-Based Simulation of Social Networks," *Proceedings of the 3rd Symposium on Social Networks and Multiagent Systems* (*SNAMAS'11*), pp.27-32, Yorkshire, UK, April 4-5, 2011.

[37] Béla Bollobás, "Random Graphs," *Cambridge University Press, 2nd Edition*, 2001.

[38] Shaomei Wu, Atish Das Sarma, Alex Frabrikant, Silvio Lattanzi and Andrew Tomkins, "Arrival and Departure Dynamics in Social Networks," *Proceedings of the 6th Web Search and Data Mining Conference (WSDM'13)*, pp.223-242,

[39] Pragnesh Jay Modi, Wei-Min Shen, Milind Tambe and Makoto Yokoo, "Adopt: Asynchronous Distributed Constraint Optimization with Quality Guarantees," *Artificial Intelligence*, 161(1-2): 149-180, 2005.

[40] Roger Mailler and Victor Lesser, "A Cooperative Mediation-Based Protocol for Dynamic Distributed Resource Allocation," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 36(1): 80-91, 2006.

[41] Yichuan Jiang, J.C. Jiang, "Diffusion in Social Networks: A Multiagent Perspective," IEEE Transactions on Systems, Man, and Cybernetics: Systems, 45(2): 198 – 213, 2015.

Wanyuan Wang (S'13) received his BS degree in information and computing science from Nanjing University of Aeronautics and Astronautics, Nanjing, China, 2011. He is currently pursuing a Ph.D. degree at the Distributed Intelligence and Social Computing Laboratory, School of Computer Science and Engineering, Southeast University. He has published several articles in refereed journals and conference proceedings, such as the IEEE Transactions on Parallel and Distributed Systems, the IEEE Transactions on Cybernetics, and the IEEE International Conference on Tools with Artificial Intelligence (ICTAI). He won the best student paper award from ICTAI'14. His main research interests include social networks and multiagent systems.

Yichuan Jiang (SM'13) received his PhD 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, School of Computer Science and Engineering, Southeast University, Nanjing, China. His main research interests include multiagent systems, social networks, social computing, and complex distributed systems. He has published more than 80 scientific 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, ACM Transactions on Autonomous and Adaptive Systems, the Journal of Parallel and Distributed Computing, the International Joint Conference on Artificial Intelligence (IJCAI), and the International Conference on Autonomous Agents and Multiagent Systems (AAMAS). He won the best paper award and best student paper award, respectively, from PRIMA and ICTAI. He is a senior member of IEEE, a senior member of CCF and CIE, a member of the editorial board of Advances in Internet of Things, an editor of International Journal of Networked Computing and Advanced Information Management, an editor of Operations Research and Fuzziology, and a member of the editorial board of the Chinese Journal of Computers.