Community-Aware Task Allocation for Social Networked Multiagent Systems

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

***Abstract*—In this paper, we propose a novel community-aware task allocation model for social networked multiagent systems (SN-MASs), where the agent' cooperation domain is constrained in community and each agent can negotiate only with its intracommunity member agents. Under such community-aware scenarios, we prove that it remains NP-hard to maximize system overall profit. To solve this problem effectively, we present a heuristic algorithm that is composed of three phases: 1) task selection: select the desirable task to be allocated preferentially; 2) allocation to community: allocate the selected task to communities based on a significant task-first heuristics; and 3) allocation to agent: negotiate resources for the selected task based on a nonoverlap agent-first and breadth-first resource negotiation mechanism. Through the theoretical analyses and experiments, the advantages of our presented heuristic algorithm and community-aware task allocation model are validated. 1) Our presented heuristic algorithm performs very closely to the benchmark exponential brute-force optimal algorithm and the network flow-based greedy algorithm in terms of system overall profit in small-scale applications. Moreover, in the large-scale applications, the presented heuristic algorithm achieves approximately the same overall system profit, but significantly reduces the computational load compared with the greedy algorithm. 2) Our presented community-aware task allocation model reduces the system communication cost compared with the previous global-aware task allocation model and improves the system overall profit greatly compared with the previous local neighbor-aware task allocation model.**

***Index Terms*—Community-aware, heuristic algorithm, multiagent systems, social networks, task allocation.**

I. Introduction

As SOCIAL networks have much relevance to many social systems and the increasing development of agent technology, social networked multiagent systems (SN-MASs) have been used as a platform to model and develop various real-world applications, such as the transportation systems [1], [2], the social economic systems [3]–[6], and the online friendship network systems (e.g., Twitter and Facebook) [7]–[9]. In general SN-MASs, agents are interconnected through social networks and tasks arrive at agents in a distributed fashion. When a complex task is initiated by an agent, the initiator agent needs to negotiate with other cooperative agents constrained by the network structure for acquiring enough resources to accomplish this complex task [10]–[12], which can be called task allocation for SN-MASs [13]–[16].

Manuscript received April 23, 2013; revised September 12, 2013; accepted October 22, 2013. Date of publication November 26, 2013; date of current version August 14, 2014. This work was supported in part by the National Natural Science Foundation of China under Grant 61170164, in part by the Program for Distinguished Talents of Six Domains in Jiangsu Province under Grant 2011-DZ023, and in part by the Funds for Distinguished Young Scholars of Jiangsu Province under Grant BK2012020. This paper was recommended by Associate Editor B. Chaib-draa.

The authors are with the Distributed Intelligence and Social Computing Laboratory, and the Key Laboratory of Computer Network and Information Integration of State Education Ministry, School of Computer Science and Engineering, Southeast University, Nanjing 211189, China (*Corresponding author: Yichuan Jiang*, e-mail: yjiang@seu.edu.cn).

Digital Object Identifier 10.1109/TCYB.2013.2289327

Generally, the global organization of many SN-MASs is constituted of communities where agents have denser and better interactions with their intracommunity members than with the intercommunity members [1]–[4], [7], [8], [17]–[19]. Moreover, most actual social systems consist of highly overlapping communities in which some nodes simultaneously belong to more than one community [9], [20] (see Fig. 1). In this paper, by being aware of the community property in SN-MASs, we propose a new variant of task allocation model for SN-MASs, where agent' cooperation domain is constrained in community and the initiator agent can negotiate only with its intracommunity member agents.

In the current work, connections represent communication possibilities and the closer the connection, the less the communication cost [6], [12], [13]. A community structure is added on top of SN-MASs to represent the set of agents that could cooperate to perform a task. This kind of task allocation model can be applicable to a variety of practical applications [1], [2], [21], [22]. For example, in the online friendships network systems, because of the high degree of similarity within social group (community), individuals always cooperate with the partners that belong to the same group for common group goals [21], [22]. Similar situations also occur in the transportation systems, the cooperation of a transportation company is always limited within these geographical closer transportation companies that locate at the same region (community) due to the time and fuel (communication cost) constraints [1], [2]. Therefore, it is essential to devise a community-aware task allocation model that considers the communication cost for SN-MASs.

Under such community-aware scenarios, to maximize system overall profit and to avoid producing heavy communication cost to the system, agents should decide which tasks to execute and how many resources to contribute for these tasks. Although there are some related algorithms [3], [5] that can deal with this community-aware task allocation problem to some extent, all of these algorithms either do not consider minimizing system communication cost that is critical to the performance of task execution in SN-MASs or cannot apply

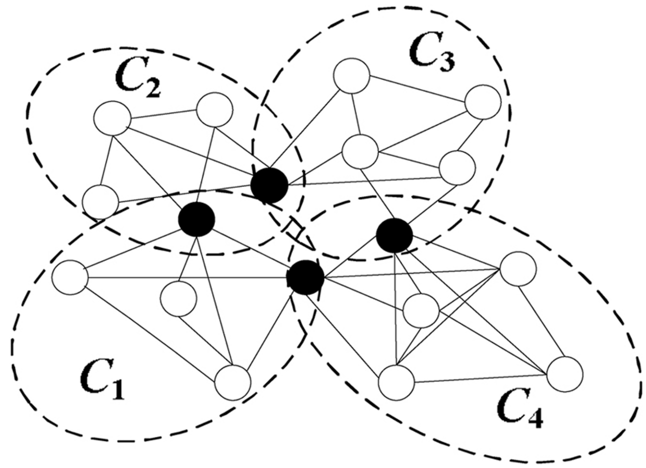

Fig. 1. Illustration of a social network with four over lapping communities; the nodes encircled by a common ellipse belong to the same community and the overlapped nodes are colored in black. For example, in the air transportation systems, each node represents a city; each edge represents a nonstop flight between two cities, and community represents locations that have closer geographical distance [1].

to large-scale applications due to the high time complexity. To solve this problem effectively, we introduce a heuristic algorithm that consists of three phases: 1) task selection: a significance-based task ranking approach is designed to ensure that the desirable tasks are allocated preferentially; 2) allocation to community: a significant task first heuristics is devised to ensure that communities contribute their redundant resources to the significant tasks first; and 3) allocation to agent: a nonoverlap agent-first and breadth-first resource negotiation mechanism is developed to ensure that the initiator agent of the selected task negotiates with the nonoverlapping agents and the agents with less communication distance first. The theoretical analyses and experiments prove that this heuristic algorithm can effectively deal with the existing problem.

To conclude, in this paper, we make the following contributions.

1) We propose a novel community-aware task allocation model for SN-MASs, where agent negotiates only with its intracommunity partners. This model is consistent with many real-world scenarios, for example, in an intergroup conflict case, all of the intragroup members are mutually beneficial if they cooperate in competing against the out group [23].
2) Under this community-aware model, we introduce a heuristic algorithm to allocate tasks to agents with the aims of reducing time complexity, yielding higher overall profit and producing less communication cost.
3) Through the theoretical analyses and experiments, the advantages of our presented heuristic algorithm and community-aware task allocation model are validated: 1) our presented heuristic algorithm performs very closely to the benchmark exponential brute-force optimal algorithm and the network flow-based greedy algorithm in terms of system overall profit in small-scale applications. Moreover, in the large-scale applications, the presented heuristic algorithm achieves approximately the same overall system profit, but significantly reduces the computational load compared with the greedy algorithm; 2) our presented community-aware task allocation model reduces the system communication cost compared with the previous global-aware task allocation model and improves the system overall profit greatly compared with the previous local neighbor-aware task allocation model.

The remainder of this paper is organized as follows. In Section II, we compare our work with the related work on this subject. In Section III, we introduce the problem and analyze its complexity. In Section IV, we propose the heuristic task allocation algorithm. In Section V, we analyze the properties of the heuristic algorithm. In Section VI, we conduct a set of experiments to evaluate the heuristic algorithm and the community-aware task allocation model. Finally, we conclude our paper and discuss our future work in Section VII.

II. Related Work

A. Task Allocation Based on Coalition Formation

Coalition formation is an important cooperation pattern in multiagent systems. Within coalitions, agents can jointly perform tasks that they would otherwise be unable to perform or would perform inefficiently [24]–[28]. For example, Manisterski *et al.* [25] proposed a centralized algorithm that utilizes a minimum weighted perfect matching technique to schedule agents to perform tasks. On the other hand, Kraus *et al.* [24] develop a decentralized auction protocol in which each agent decides to accept or reject coalitions based on a heuristic coalition rank strategy. Shehory and Kraus [26] and Michalak *et al.* [27] also study the distributed algorithms with low ratio bounds, low computation, and communication complexities for forming efficient coalitions. In [27], they achieve the optimal solution through distributing the necessary calculation burden among agents. Another distributed mechanism is to utilize a mediator agent to allocate tasks to server agents based on a gradient ascent learning algorithm with aim of minimizing the average turn-around time [28]. However, few of these approaches assume the existence of a social network environment. In contrast to these studies, we consider a new variant of the task allocation problem in SN-MASs.

B. Task Allocation for Social Networked Multiagent Systems (SN-MASs)

The related researches on task allocation for SN-MASs can be categorized into the global-aware task allocation model and the local neighbor-aware task allocation model according to the domain of partners the initiator agent negotiates with.

In the global-aware task allocation model, agents can cooperate with all of the other agents in the system [12]–[16]. For example, in [13]–[16], the researchers propose a contextual resource negotiation-based task allocation mechanism, where the central controller first allocates a task to a manager agent who owns rich contextual resource, and then the manager negotiates with all of its contextual agents for acquiring enough resources to accomplish the allocated task. Kota *et al.* [12] proposed a distributed self-adaption method for dynamic agent organizations. In this method, when a task is submitted to a superior agent and this superior cannot perform this task individually, the superior agent can delegate this task to all of its subordinate or the peer agents. However, in reality, it is impracticable for each agent to consider all of the other agents to cooperate with [11]. For example, in the transportation systems, a transportation company prefers to cooperatively work together with these geographical closer transportation companies due to the time and fuel constraint. In contrast to [12]–[16], we constrain the cooperation domain of agents within communities.

In the local neighbor-aware model, agents cooperate only with their immediate neighbors [3], [5]. At the first glance, it seems that there are several similarities between our work and the work in [3] and [5]. For example, we both constrain the agent' cooperation domain with limited partners and we both strive to devise desirable algorithms to maximize system overall profit. However, there are two important features that distinguish our model from the previous model, show as follows.

1) First, in the local neighbor-aware model [3], [5], for a given agent a_i, it is feasible for a_i to allocate its tasks to any other agents that have direct interaction with a_i. However, in our community-aware model, the cooperation with a direct neighbor is not allowed if the particular neighbor belongs to a different community with a_i's. Moreover, in our model, because of the different social distances among community members, agents need to decide which community partners to cooperate with to minimize the communication cost. While in [3] and [5], because of the social distances among any pairwise cooperators are the same (i.e., one hop social distance), agents do not necessarily need such a complex negotiation process.
2) Second, in [3] and [5], the researchers propose a network-based greedy algorithm to achieve the approximate optimal solution. In the greedy algorithm, a central controller utilizes the network flow technique to allocate tasks in decreasing order of their efficiency values. However, their algorithm neither considers minimizing system communication cost that is critical to the performance of task execution in SN-MASs nor can apply to large-scale applications due to the high time complexity. In contrast to these work, we propose an effective heuristic algorithm with a lower computation complexity and less overhead communication costs. The proposed algorithm considers the community-aware task allocation characteristics (e.g., social position of agents and tasks, task profitability, community size, etc.) to guarantee agents to execute the desirable tasks preferentially, communities to contribute their redundant resources to the significant tasks first, and agents to cooperate with nonoverlapping agents and the agents with less communication distance first.

III. Problem Description and Analysis

A. *Formalization of Community-Aware Task Allocation*

In general social networks, communities are subgroups of nodes that interact more with nodes in the same group but far less with nodes in different groups [29]. Moreover, most actual social networks consist of highly overlapping communities in which some nodes simultaneously belong to more than one community [20]. Now, we will give the definition of the community-aware social networked multiagent systems (see Table I for a summary of notations used in this paper).

Definition 1: Community-Aware Social Networked Multiagent Systems (*CA-SN-MASs*). A community-aware social networked multiagent system is defined as *CA-SN-MAS* = $<A, E, C>$, where $A = \{a_1, a_2, \ldots, a_m\}$ is the set of agents, $\forall(a_i,a_j) \in E$ indicates the existence of a social interaction between agents a_i and a_j, and $C = \{C_1, C_2, \ldots, C_q\}$ indicates the overlapping communities that constitute the network. Each agent $a_i \in A$ is aware of its community attribute, in other words, which community it belongs to and who its community partners are. In turn, any two communities C_i and C_j might share a set of overlap agents $ov(C_i, C_j)$, and any two communities C_i and C_j are adjacent if and only if there exists overlap between them.

We utilize the measure overlap degree to quantify the community structure of a *CA-SN-MAS*.

Definition 2: Overlap Degree. The size of a *CA-SN-MAS* is given by the number of agents in that network, and the size of a community in the *CA-SN-MAS* is given by the number of agents in that community. Then, the overlap degree of the *CA-SN-MAS*, *overlap*, is defined as the sum of all of the communities' sizes divided by the size of the *CA-SN-MAS*.

Let there be k resource types $R = \{r_1, r_2, \ldots, r_k\}$ available in a *CA-SN-MAS* = $<A,E,C>$. Then, each agent $a_i \in A$ can be defined by a two-tuple $<rsc(a_i), com(a_i)>$, where the resource function $rsc(a_i): A \times R \rightarrow \mathbb{N}$ indicates the amount of each resource type owned by agent a_i (here, we assume that a unit resource cannot be used more than one time), and the community function $com(a_i)$: $A \rightarrow C$ indicates the community(ies) that a_i belongs to.

We consider a set of tasks $T = \{t_1, t_2, \ldots, t_n\}$ initiated by agents in a decentralized fashion. Then, each task $t_j \in T$ can be defined by a three-tuple $<req(t_j), p(t_j), ini(t_j)>$, where the resource function $req(t_j)$: $T \times R \rightarrow \mathbb{N}$ indicates the amount of each resource type required by t_j. The payment function $p(t_j)$: $T \rightarrow \mathbb{R}$ indicates the reward the system will obtain if t_j is completed successfully (a task is completed successfully if and only if all of its required resources are satisfied and partial fulfillment of a task yields no payment). The initiator function $ini(t_j)$: $T \rightarrow A$ indicates the initiator agent where task t_j is initiated.

Under this community-aware scenario, each initiator agent a_i can negotiate only with the partners that belong to the same community with a_i's. We summarize a resource access function δ between agent and task

$$\delta(a_i, ini(t_j)) = \begin{cases} 1 & if\ a_i\ and\ ini(t_j)\ belong\ to\ the\ same \\ & \qquad community, \\ 0 & otherwise. \end{cases} \tag{1}$$

Given an agent a_i and a task t_j, if $\delta(a_i, int(t_j)) = 1$, then the initiator of task t_j can cooperate with a_i for executing t_j; otherwise, it cannot.

A task allocation is defined as the mapping function Φ: $A\times T \times R\rightarrow \mathbb{N}$, which indicates the amount of each resource type that agents contribute to tasks. An allocation Φ is feasible if and only if it satisfies the community consensus, that is, each task must be executed by the agents that belong to the community including the task's initiator.

Definition 3: Social Welfare. Let $CS(\Phi)$ be the set of tasks completed successfully by an allocation Φ. Then, the social welfare of this allocation Φ, $SW(\Phi)$, is given by the sum of the payments of the tasks that completed successfully, i.e., $SW(\Phi) = \sum_{t\in CS(\Phi)} p(t)$.

Definition 4: Community-Aware Task Allocation Problem. Given a community-aware social networked multiagent system $CA\text{-}SN\text{-}MAS = <A, E, C>$ and a set of tasks $T = \{t_1, \ldots, t_n\}$ initiated by the agents A independently. The community-aware task allocation problem (*CA-TAP*) is to find the optimal feasible allocation Φ^* with the highest social welfare, in other words, for any other feasible task allocation Φ', $SW(\Phi^*) \geq SW(\Phi')$.

Example 1: Fig. 2 shows an instance of the *CA-TAP*. In Fig. 2, we assume that there are two resource types, r_1 and r_2, available in the *CA-TAP* and there are six agents $a_1 = <\{3r_1\},\{C_1\}>, a_2 = <\{6r_2\},\{C_1\}>, a_3 = <\{3r_1,4r_2\},\{C_1,C_2\}>, a_4 = <\{3r_1,2r_2\},\{C_2\}>, a_5 = <\{2r_1,3r_2\},\{C_2\}>$ and $a_6 = <\{3r_1,1r_2\},\{C_2\}>$ residing in the two overlapping communities C_1 and C_2. Now, there are three tasks $t_1 = <\{4r_1,6r_2\}, 8,a_2>$, $t_2 = <\{10r_1,10r_2\},16,a_3>$, and $t_3 = <\{8r_1,6r_2\},10,a_4>$ submitted to agents a_2, a_3 and a_4, respectively. Obviously, the initiator agent a_2 lacks the necessary resources $\{4r_1\}$ to execute its initiated task t_1, so it has to negotiate with its community partners (i.e., a_1 and a_3) for assistance. If a_2 achieves enough resources from its community partners (e.g., a_1 provides $\{3r_1\}$ to a_2, and a_3 provides $\{1r_1\}$ to a_2) for performing t_1, the system will obtain payment $p(t_1) = 8$. However, a_3 would then lack more resources to execute its own local task t_2. Analogously, a_3 will have to acquire the lacked resources from its community partners $\{a_1,a_2,a_4,a_5,a_6\}$. When a_3 negotiates with a_4, a_4 should be cautious to provide the necessary resources to $a_3$for three reasons: 1) a_4 is also responsible for its own local task t_3, in other words, if it contributes the necessary resources to a_3, then it might not be able to accomplish t_3 anymore; 2) a_3 owns more community partners than a_4(a_4's community partners are a_3, a_5 and a_6), in other words, if a_4 does not provide the necessary resources to a_3,a_3 can still procure enough resources from others; and 3) task t_2 has a higher payment than t_3. The situation becomes more complex if the numbers of agents, tasks, resource types, and communities become larger.

B. Complexity Analysis of the Community-Aware Task Allocation Problem

It is worth noting that one of the key subproblems of the community-aware task allocation problem is to allocate overlap agents to tasks optimally. In this section, we will prove that this subproblem is NP-hard because NP-hard 3-SATISFIABILITY (3-SAT) is reducible to it.

In a 3-SAT Boolean formula $\sum$, $\sum$ is expressed as an AND of clauses in which a clause is the OR of exactly three distinct

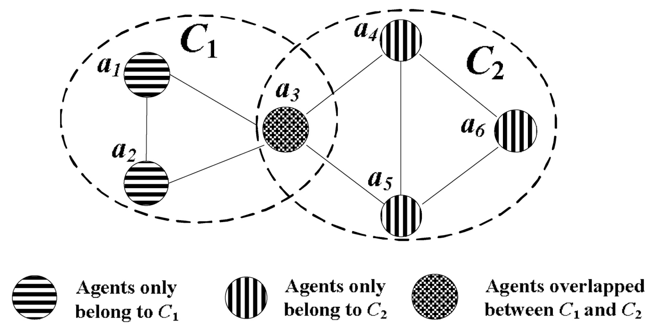

Fig. 2. Example instance for illustrating the community-aware task allocation problem.

literals. Here, a literal in a clause is an occurrence of a variable l or its complement $\neg l$. The 3-SAT problem is to determine whether there is an assignment (TRUE or FALSE) to the variables that makes the formula evaluate to TRUE or not [30]. If yes, the formula $\sum$ is a satisfiable formula; otherwise, it is an unsatisfiable formula. For example, the following formula $\sum = (l_1\vee\neg l_2\vee\neg l_3) \wedge (l_2\vee l_3\vee\neg l_4) \wedge (\neg l_1\vee l_2\vee l_4)$ is a satisfiable 3-SAT formula, and a satisfying assignment is $<l_1 = 0, l_2 = 1, l_3 = 0, l_4 = 1>$.

Let $\sum = C_1\wedge C_2\wedge\ldots\wedge C_k$ be the input for the 3-SAT problem, where C_r is a clause with three literals $(l_{rm}\vee l_{rn}\vee l_{rp})(1\leq r\leq k$, m, n, p are arbitrary integers). By referring to related work on generating the maximal independent set of a graph in [31], we next construct a graph $G(V,E)$ from $\sum$ for the community-aware task allocation problem. For each clause C_r and each literal l_{ri}in $\sum$, we create a clause vertex vc_r and a literal vertex vl_{ri} in G, correspondingly. For each pairwise contradiction literals $(l_{ri}, \neg l_{si})$ in $\sum$, $1\leq r,s\leq k$, we create an auxiliary vertex $va_i^{r,s}$ in G, correspondingly. Then, in graph G, we make each clause vertex vc_r adjacent to all literal vertexes vl_{ri} that it contains and each pairwise contradiction literal vertexes vl_{ri} and $\neg vl_{si}$ adjacent to their generated auxiliary vertex $va_i^{r,s}$. These are all of the vertexes V and edges E of G. Specifically, we regard the literal vertex as the community in graph G, and then, we assign each community vl_{ri}a task t_{ri} with one unit of payment, and the required resources of t_{ri} are its neighboring vertexes, i.e., $t_{ri} = <\{v|(vl_{ri}, v)\in E\}, 1, vl_{ri}>$. This indicates that the clause vertex vc_r simultaneously belongs to the three communities $\{vl_{rm}, vl_{rn}, vl_{rp}\}$ that it contains, and the auxiliary vertex $va_i^{r,s}$ is the overlapping vertex shared by the pairwise contradiction community vl_{ri} and $\neg vl_{si}$. Up to this point, we have constructed the CA-TAP from the 3-SAT formula $\sum$ (An instance of the construction is shown in Fig. 3), and the construction of the community-aware task allocation problem from an arbitrary 3-SAT problem takes polynomial time. Next, we will show that the constructed CA-TAP has a task allocation with k unit payments if and only if $\sum$ is satisfiable.

Suppose that $\sum$ has a satisfying assignment. Then, each clause C_r has at least one literal $l_{ri} = 1$and its corresponding vertex in G is vl_{ri}. Selecting one such literal from each clause, we obtain the corresponding k literal vertices in G, which stand for the k communities of the CA-TAP that are denoted as $S = \{vl_{1i}, vl_{2j}, \ldots, vl_{kq}\}$. Then, any two of the k communities

TABLE I
DEFINITIONS OF NOTATIONS

Notation	Definition
a_i	Agent i
t_j	Task j
C_q	Community q
r_k	Resource type k
$W(t)$	The sum of the resources that task t requires: $W(t)=\sum_{i=1}^{k} req(t,r_i)$
$com(a_i)$	The community(ies) that a_i belongs to
$A(C_q)$	The agents reside in community C_q: $A(C_q)=\{a_i \mid C_q \in com(a_i)\}$
$T(C_q)$	The tasks reside in community C_q: $T(C_q)=\{t_j \mid C_q \in com(ini(t_j))\}$
$\Omega(t_j)$	The accessible agents of task t_j: $\Omega(t_j)=\{a_i \mid com(a_i) \cap com(ini(t_j)) \neq \emptyset\}$
$ov(C_i,C_j)$	The overlap agents overlapped between communities C_i and C_j
$res(a_i,r_k)$	The amount of resource type r_k owned by a_i
$req(t_j,r_k)$	The amount of resource type r_k required by t_j
$nov\text{-}A(C_q)$	The non-overlap agents of community C_q
$ar(t_j,r_k)$	The amount of accessible resources of resource type r_k for task t_j: $ar(t_j,r_k)=\sum_{a\in\Omega(t_j)} rsc(a,r_k)$
$cr(a_i,t_j)$	Resource contribution of agent a_i for task t_j
$uT(C_q)$	The unallocated tasks of community C_q
$Rr(C_q,T^*,r_k)$	Community C_q's redundant resources of resource type r_k after executing a set of tasks T^*
$\theta(C_q,r_k)$	Non-overlapping agents' remaining resources of resource type r_k of community C_q
$\varphi(C_q,r_k)$	The resources of resource type r_k of community C_q that have been used

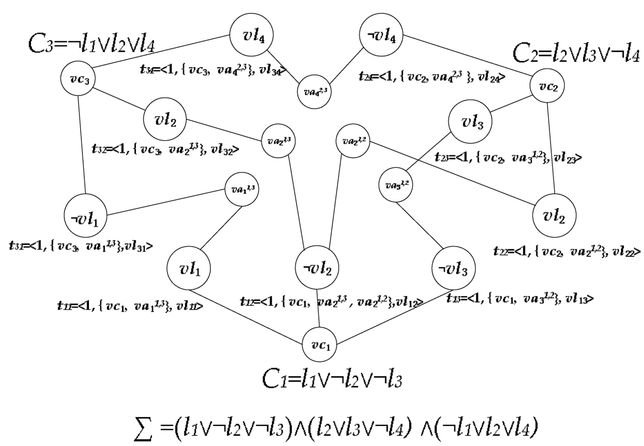

Fig. 3. Example instance for showing the construction from the 3-SAT to the CA-TAP.

vl_{ri} and vl_{sj} in S do not share an overlapping auxiliary vertex $va_i^{r,s}$ because any literal and its complement cannot be both assigned to 1. Therefore, those k tasks residing in the k communities can be accomplished successfully, which results in a total of k unit payments.

Conversely, if there is an allocation for the CA-TAP with a total of k unit payments, where the allocated tasks' located communities are denoted as $S = \{vl_{1i}, vl_{2j}, \ldots, vl_{kq}\}$, then any two of the literals in S must belong to different clauses because the overlapping clause vertex vc_r cannot be allocated to more than one community task (i.e., t_{rm},t_{rn} and t_{rp}). Additionally, if $vl_{ri} \in S$, then its complement vertex $\neg vl_{si} \notin S$ because vl_{ri} and $\neg vl_{si}$ share the overlapping auxiliary vertex $va_i^{r,s}$. Thus, we can assign 1 to the corresponding k literals $\{l_{1i}, l_{2j}, \ldots, l_{kq}\}$ in $\sum$. In other words, $l_{ri} = 1$ if $vl_{ri} \in S$. If there are other literals in $\sum$ that are not assigned yet, then we arbitrarily assign each of them the value 1 and its complement the value 0.

Obviously, this assignment satisfies the formula $\sum$. Now, we can determine that 3-SAT is reducible to the subproblem of allocating overlap agents to communities, which in turn proves that the community-aware task allocation problem is NP-hard.

IV. Task Allocation Algorithm

Recall the community-aware task allocation problem defined in Definition 4: given a set of networked agents and a set of tasks initiated by these agents, we consider an approach to allocate each task to a group of agents that must satisfy the constraint of community structure. Under the limit of cooperation among intracommunity partners, to maximize social welfare and to avoid producing heavy communication cost, agents should decide which tasks to execute and how many resources to contribute for these tasks. To solve such an NP-hard problem effectively, we present a heuristic algorithm that can be implemented through the following three phases:

1) Task selection: we rank tasks with respect to a significance measure and allocate them in order of their ranking (Section IV-A);
2) Allocation to community: for each community that the selected task belongs to, we calculate its resource contribution to this selected task (Section IV-B);
3) Allocation to agent: for each community that makes contribution to the selected task, the initiator of the selected task negotiates with the agents in that community to procure the contributed resources (Section IV-C).

A. Task Selection

In economics, the metric of profitability is always used as an important index for a firm stake holder's decision making, which is given by the ratio between a firm's annual income and its annual capital investment [32]. Such an idea can be introduced in the measure of a task's profitability. Next, by referring to the related definition in [3], we present the concept of a task's profitability.

Definition 5: Task Profitability. Let $W(t)$ denote the sum of the resources that task t requires. The profitability, $pro(t)$, of task t is defined as the payment $p(t)$ of t divided by $W(t)$, i.e., $pro(t) = p(t)/W(t)$.

A task that has a higher profitability indicates that either the task possesses a larger payment or the task requires fewer re sources, or both; the system will achieve a higher efficiency if this kind of task is completed preferentially. However, the above heuristics considers only a task's payment and resource properties, ignores the task's fitness to its residing community. For example, now there are three tasks $t_1 = <\{4,6\},8,a_i>$, $t_2 = <\{8,6\},10,a_j>$, $t_3 = <\{15,15\},20,a_k>$ submitted to agents a_i, a_j and a_k, respectively, where $pro(t_1)>pro(t_2)>pro(t_3)$. Only by the profitability heuristics, tasks t_1 and $t_2$will be executed preferentially. In case that after executing t_1 and t_2, task t_3 cannot be satisfied anymore, the system will obtain 18 unit payments. Alternatively, if the system completes task t_3 successfully before executing tasks t_1 and t_2, it will obtain at least 20 unit payments. Therefore, it is also necessary to be aware of a task with a large payment that lies in somewhere where there are sufficient resources to access.

Definition 6: Task Fitness. The accessible resources of a task t, $ar(t)$**,** is defined as the sum of available resources of these agents that belong to the community including the task's initiator. The fitness of t, $fitness(t)$, then, is defined as

$$fitness(t) = 1 - \frac{\sum_{i=1}^{k} \max(ar(t, r_i) - req(t, r_i), 0)}{\sum_{i=1}^{k} ar(t, r_i).} \tag{2}$$

Finally, we combine the two factors of task profitability and task fitness and make a tradeoff between them to introduce the concept of task significance.

Definition 7: Task Significance. Let $pro(t)$ and $fitness(t)$ denote the profitability and fitness of task t, respectively. The significance of t, $sig(t)$, is defined as

$$sig(t) = \alpha \cdot pro(t) + (1 - \alpha) \cdot fitness(t) \tag{3}$$

where α is the parameter within the closed interval [0,1]. This parameter α is used to determine the relative importance of the two measures of a task.

The heuristic algorithm ranks the tasks in order of descending significance first and then allocates these tasks to communities in turn.

B. Allocation to Community

We are mainly concerned with the social position of agents and tasks in social networks when addressing the allocation problem. The social position of agents and tasks can be categorized into nonoverlap and overlap.

Definition 8: Social Position. Given a $CA\text{-}SN\text{-}MAS = <A, E, C>$ and a set of tasks $T = \{t_1, \ldots, t_n\}$ initiated by these agents A. An agent $a_i \in A$ is denoted as a nonoverlap agent if and only if it belongs to one community and the task $t_j \in T$ initiated by a_i is called a nonoverlap task; the other agents and tasks are called overlap agents and tasks, respectively.

Algorithm 1: Resource Negotiation(t_j,a_i,$flag$,C_q,R_{cont})(RN)

t_j: the task to be executed;
a_i: the initiator agent of t_j where RN is implemented;
$flag$=0: a_i negotiates with the non-overlap agents; $flag$=1: a_i negotiates with the overlaps; /*Here, we assume $flag$=0*/
C_q: the community that makes a contribution to t_j and R_{cont} is its resource contribution for t_j.

1. Set thetags for the agents in community C_q to 0 and the tags for agents in other communities to 1.
2. Create Queue(Q);
3. Insert Queue(Q,a_i); set the tag of a_i to 1.
4. While(!isEmpty(Q)&&R_{cont}≠{})**do**
5. a_{out}=**Out** Queue(Q),
6. **If**(|$com(a_{out})$|==1), **then**
7. R'=R_{cont};R_{cont}=R_{cont}-$rsc(a_{out})$; $rsc(a_{out})$=$rsc(a_{out})$-(R'-R_{cont}).
8. **End if**
9. **For** $\forall a_{adj} \in adj(a_{out})$, **do** /*$adj(a_{out})$ is the set of agents that are adjacent to a_{out}. */
10. **If** tag of a_{adj} is 0, **then**
11. **Insert** Queue(Q, a_{adj}); set the tag of a_{adj} to 1.
12. End.

Now, we are ready to introduce the task allocation to community mechanism. This mechanism can be divided into rounds and it ends when there are no more tasks to be allocated. In each round, we sort the remaining unallocated tasks in decreasing order of significance and allocate the first task that with the maximum significance value. Without loss of generality, in a certain round, we assume the sorted tasks are $T' = \{t_1', t_2', \ldots, t_n'\}$, and the current task to be allocated is t_1'. The process of allocating t_1' to communities consists of the following two stages.

Checking stage: If task t_1'can be satisfied by the remaining resources of its accessible agents $\Omega(t_1')$, we will allocate t_1' to its accessible communities in the next stage. Otherwise, task t_1' will be removed from the system.

Allocation stage: In this stage, we adopt different strategies to allocate t_1' according to its social position.

1) *Nonoverlap case*: Task t_1' is a nonoverlap task, then we will allocate t_1' to the community $com(ini(t_1'))$ it belongs to.
2) *Overlap case*: Task t_1' is an overlap task, the allocation to community process then can be implemented through the following three steps:
    a) Allocation to a resource-rich community: If there exists a resource-rich community C_q that t_1' belongs to (i.e., $C_q \in com(ini(t_1'))$), it can satisfy all of its unallocated tasks $uT(C_q)$ by only its nonoverlap agents $nov\text{-}A(C_q)$, then we will allocate t_1' to this resource-rich community. Otherwise, go to Step b).
    b) Allocation to community's redundant resources: For each community C_q that t_1' belongs to, it calculates its redundant resources

Algorithm 2: The Heuristic Algorithm (λ) /*λ is the set of tasks that are completed successfully by the heuristic algorithm*/

1. **Create** ArrayList(L) and insert all the tasks $t \in T$ into L.
2. **While** (!isEmpty(L))**do**
3. **Sort** the tasks in L in order of decreasing significance, i.e., $sig(L[0]) \geq sig(L[1]) \ldots \geq sig(L[L.\text{size}()\text{-}1])$.
4. **Set** t_{out}=L[0] and delete the task t_{out} from L.
5. **If** t_{out} can be satisfied by its accessible agents(*i.e.*,$\forall r \in R, req(t_{out}, r) \leq \sum_{a \in \Omega(t_{out})} rsc(a,r)$), **then**
6. **If** t_{out} is a non-overlap task, **then**
7. *RN*(t_{out}, $ini(t_{out})$, 0,$com(ini(t_{out}))$, $req(t_{out})$);*RN*(t_{out}, $ini(t_{out})$,1,$com(ini(t_{out}))$, $req(t_{out})$). /* *The second time RN is called by* $ini(t_{out})$ *for negotiating with the overlaps,* $ini(t_{out})$*only negotiates the resources for its remaining resource requirement**/
8. **Else**
9. **For**$\forall C_q \in com(ini(t_{out}))$ **do**
10. **If** the community C_q can perform all of its unallocated tasks by its non-overlap agents, **then**
11. *RN*(t_{out} , $ini(t_{out})$, 0, C_q, $req(t_{out})$); go to Step 23.
12. **End for**
13. **For** $\forall C_q \in com(ini(t_{out}))$**do**
14. **Calculate** its redundant resources: $\forall r \in R, Rr(C_q, uT(C_q) \backslash \{t_{out}\}, r) = \sum_{a \in A(C_q)} rsc(a,r) - \sum_{t \in uT(C_q) \backslash \{t_{out}\}} req(t,r)$.
15. *RN*(t_{out}, $ini(t_{out})$,0,C_q, $Rr(C_q, uT(C_q) \backslash \{t_{out}\})$); *RN*(t_{out}, $ini(t_{out})$, 1, C_q, $Rr(C_q, uT(C_q) \backslash \{t_{out}\})$).
16. **End for**
17. **If** t_{out} has not been accomplished, **then**
18. **Initialize** release_taskId=L.size()-1.
19. **While** ($req(t_{out}) \neq \emptyset$)**do**
20. **Release** the task L[release_taskId]; repeat step 13~16; release_taskId=release_taskId-1.
21. **End if**
22. **End else**
23. **Update** $\lambda \leftarrow \lambda \cup \{t_{out}\}$.
24. **End if**
25. **End while**
26. **Return** λ and the task allocation result.

$Rr(C_q, uT(C_q) \backslash \{t'_1\})$ after executing the other unallocated tasks $uT(C_q) \backslash \{t'_1\}$ residing in C_q: $\forall r \in R$, $Rr(C_q, uT(C_q) \backslash \{t'_1\}, r) = \sum_{\alpha \in A(C_q)} rsc(a,r) - \sum_{t \in uT(C_q) \backslash \{t'_1\}} req(t,r)$, (here, we just aim at computing the redundant resources rather than really executing these unallocated tasks) and contributes the redundant resources for t_1'. If the sum of the redundant resources of all of the accessible communities are not sufficient for t_1', go to Step c).

c) Allocation to the free resources by releasing the tasks with lower significance values: Obviously, the reason that task t_1'cannot yet be satisfied is that the other unallocated tasks occupy the critical resources that t_1' requires as well. Thus, we must free the resources occupied by these unallocated less significant tasks. Here, we release the unallocated tasks one by one according to the inverted order of the sorted list T', that is from task t_n' to task t_2'. Each time we release a less significant task, we repeat Step b). We iterate this releasing step until t_1' be satisfied.

C. Allocation to Agent

After calculating the resource contribution of a community to a selected task, the initiator of this task should negotiate with the partners of that community to procure the contributed resources with aims of both improving system efficiency and reducing communication cost.

To improve system efficiency, we present a *nonoverlap agent first* (*NAF*) heuristic: for a task t, let there be a set of agents Δ in which an overlap agent a_i is included that can satisfy t's resource requirement. Now there exists another agent set $\triangle *$ where the overlap agent a_i is excluded that also satisfies all of the resources required by t. We suggest the nonoverlap agents set $\triangle *$ executing the task.

To reduce communication cost, we utilize the distributed *breadth first* (*BF*) resource negotiation approach. In this negotiation approach, the initiator agent negotiates with its community partners from nearby to far-away until the requested resources are satisfied [13]. Notice that when an agent agrees to cooperative with an initiator agent, it offers all of its remaining available resources to this initiator. The initiator's resource negotiation process is outlined in Algorithm 1.

D. Heuristic Algorithm and Case Study

With the above discussion, a formal description of the heuristic algorithm can be seen in Algorithm 2, and to illustrate the proposed heuristic algorithm clearly, we take example 1 as a concrete case to study.

Example 1 (continue). As discussed before, the heuristic algorithm consists of the following three phases:

[Phase 1] Task selection: According to the significance ranking criterion, we have $sig(t_2)$>$sig(t_1)$>$sig(t_3)$ [$sig(t_2) = \alpha \cdot pro(t_2) + (1\text{-}\alpha) \cdot fitness(t_2)$=0.8×(16/20)+0.2×[1-((14-10)+(16-10))/(14+16)] = 0.77>$sig(t_1)$ = 0.765>$sig(t_3)$ = 0.71, we set α=0.8 in (3)]. Then the first task to be allocated is t_2.

[Phase 2] Allocation to community: Task t_2 can be allocated through the following two stages:

[Phase 2.1] Checking stage: It is easy to check that t_2 can be satisfied by its accessible agents (for r_1 of task t_2: $req(t_2, r_1) = 10 < \sum_{a\in\{a_1,\ldots,a_6\}} rsc(a, r_1) = 14$, for r_2 of task t_2: $req(t_2, r_2) = 10 < \sum_{a\in\{a_1,\ldots,a_6\}} rsc(a, r_2) = 16$), then we will allocate t_2 to its accessible agents in the allocation stage.

[Phase 2.2] Allocation stage: Because of the overlap social position of task t_2, the allocation process for t_2 then can be implemented through the following three steps:

Step 1: Allocation to a resource-rich community: Neither community C_1 nor C_2 can accomplish its unallocated tasks only by its nonoverlap agents (for r_1 of community C_1: $\sum_{a_1,a_2} rsc(a_i, r_1) = 3 < \sum_{t_1,t_2} req(t_i, r_1) = 14$; for r_1 of community C_2: $\sum_{a_4,a_5,a_6} rsc(a_i, r_1) = 8 < \sum_{t_2,t_3} req(t_i, r_1) = 18$). Then, go to Step 2.

Step 2: Allocation to community's redundant resources: Community C_1 computes its redundant resources $Rr(C_1,t_1) = \{2r_1,4r_2\}$ after executing its unallocated task $t_1(\sum_{a_1,a_2,a_3} rsc(a_i, r_1) - req(t_1, r_1) = 2$, $\sum_{a_1,a_2,a_3} rsc(a_i, r_2) - req(t_1, r_2) = 4)$ and contributes these redundant resources $\{2r_1,4r_2\}$ to t_2. Go to allocation to agent phase (i.e., **[Phase 3.1]**). Now task t_2's remaining required resources are: $t_2' = \{8r_1,6r_2\}$. Community C_2 computes its redundant resources $Rr\ (C_2,t_3) = \{3r_1,4r_2\}$ after executing its unallocated task t_3 and contributes the redundant resources $\{3r_1,4r_2\}$ to t_2. Go to **[Phase 3.2]**. Now, the remaining resource requirements of t_2 become $t_2'' = \{5r_1,2r_2\}$, which indicates that $t_3$has not been satisfied. Then, go to Step 3.

Step 3: Allocation to the free resources by releasing the tasks with lower significance values: Based on the lower significance task released first heuristics, the first task to be released is $t_3(sig(t_1) > sig(t_3))$. Now, the redundant resources of C_2 are $Rr(C_2,\{\}) = \{8r_1,6r_2\}$, which is sufficient for t_2''. Therefore, in this step, C_2 will contribute resources $\{5r_1,2r_2\}$ $(min(t_2'',Rr(C_2,\{\})))$ for t_2. Go to **[Phase 3.3]**.

[Phase 3] Allocation to agent:

[Phase 3.1]: According to the *NAF* and *BF* resource negotiation mechanism, the initiator agent a_3 of t_2 negotiates with the partners in community C_1 in order of $a_1 \succ a_2 \succ a_3$ (the notation ($\succ$) means the prior relationship). Then, agent a_1 contributes resources $cr(a_1,t_2) = \{2r_1\}$ and a_2 contributes resources $cr(a_2,t_2) = \{4r_2\}$ for t_2. After this contribution process, the remaining resources of agents a_1 and a_2 are: $a_1' = \{1r_1\}$ and $a_2' = \{2r_2\}$.

[Phase 3.2]: Agent a_3 negotiates with the partners in community C_2 in order of $a_4 \succ a_5 \succ a_6 \succ a_3$. Then, the partners resource contributions are: $cr(a_4,t_2) = \{3r_1,2r_2\}$ and $cr(a_5,t_2) = \{2r_2\}$. After this contribution process, the remaining resources of agents a_4 and a_5 become: $a_4' = \{\}$ and $a_5' = \{2r_1,1r_2\}$.

[Phase 3.3]: Similar to [**Phase 3.2**],a_3 accesses resources for t_2 from agents a_4, a_5, a_6, and a_3 in turn. The partners resource contributions are $cr(a_5,t_2) = \{2r_1,1r_2\}$ and $cr(a_6,t_2) = \{3r_1,1r_2\}$. Agents a_5 and a_6 update their remaining available resources: $a_5'' = \{\}$ and $a_6' = \{\}$.

Due to space limitations, the sub sequential location processes for t_1 and t_3 are not described here.

V. Properties of Heuristic Algorithm

A. Quality Guarantee Analysis

By referring to the related work in [5], we provide the worst performance ratio of the heuristic algorithm.

Theorem 1: For a given $CA-TAP$, suppose that the maximum number of resources of a task is M and the minimum payment of a task is U. Then, the heuristic algorithm has the worst performance ratio of $M(1 + (n-1)(1-\alpha)/\alpha U) + 1$, where n and α are the task number and the tradeoff factor in (3), respectively.

Proof: In the worst case, the heuristic algorithm selects only the most significant task t^* to execute, and then all of the other n-1 tasks (T-$\{t^*\}$) cannot be satisfied any more, while the optimal solution is completing all of the tasks $T_{Opt} = T$ successfully. In this worst case,[1] we can derive that the number of resources that task t^* requires is at least n-1 (if the number of resources task t^* requires is less than n-1, it cannot prevent all of the other n-1 tasks (T-$\{t^*\}$) being unallocated). Based on the fact that t^* has a higher significance value than any other task $t' \in T$-$\{t^*\}$, we have

$$\begin{aligned} sig(t') \le sig(t^*) &\Rightarrow \alpha \cdot pro(t') + (1-\alpha)\cdot fitness(t') \le \alpha \cdot pro(t^*) \\ &\quad + (1-\alpha)\cdot fitness(t^*) \\ &\Rightarrow \alpha \cdot (pro(t') - pro(t^*)) \le (1-\alpha)\cdot (fitness(t^*) \\ &\quad - fitness(t')) \\ &\Rightarrow pro(t') - pro(t^*) \le (1-\alpha)/\alpha. \end{aligned} \tag{4}$$

The inequality (4) follows from $0 < fitness(t) \le 1$ for any task t. Let $W(t)$ denote the sum of resources required by task t, then we have

$$\begin{aligned} p(t')/W(t') - p(t^*)/W(t^*) \le (1-\alpha)/\alpha \ &\Rightarrow p(t') \\ /W(t') \le p(t^*)/(n-1) + (1-\alpha)/\alpha& \\ &\Rightarrow p(t') \le M \cdot (p(t^*)/(n-1) + (1-\alpha)/\alpha). \end{aligned}$$

Let $SW(Heu)$ and $SW(Opt)$ denote the social welfare of the heuristics and that of the optimal, respectively. Then, the worst performance ratio between the optimal solution and the heuristic solution on social welfare satisfies

$$\begin{aligned} \frac{SW(Opt)}{SW(Heu)} = \frac{\sum_{t\in T_{Opt}} p(t)}{p(t^*)} &\le \frac{p(t^*) + \sum_{t\in T_{Opt}-\{t^*\}} M(\frac{p(t^*)}{n-1} + \frac{1-\alpha}{\alpha})}{p(t^*)} \\ &\le M\left(1 + \frac{(n-1)(1-\alpha)}{\alpha U}\right) + 1. \end{aligned}$$

■

B. Complexity Analysis

Besides comparing the performance of the heuristic algorithm with other algorithms on social welfare, the quality of the heuristics should also be evaluated with respect to the computational complexity.

[1]The worst case is relative to the optimal case and obviously, the optimal case is that all the tasks are completed successfully.The social welfare of the proposed worst case is $p(t*)$. Suppose that after executing task $t*$, there is another task t' that can be satisfied by the heuristics, the social welfare will become $p(t*)$+$p(t')$ which is greater than $p(t*)$. Therefore, the worst case is executing only the most significant task $t*$ and the use of resources of this task $t*$ prevents all other tasks from being satisfied.

Theorem 2: Given a *CA-TAP* with m agents, n tasks, k resource types, and q communities, the computational complexity of the heuristic algorithm is $O(mn^2kq)$.

Proof: In Algorithm 2, for Step 3, calculating the significance values of the remaining unallocated n tasks takes $O(mnk)$ computations and sorting these tasks in decreasing order of significance by a heap sort procedure takes $O(n\log(n))$ operations. For Step 5, a total of $O(mk)$ computations are used to check whether the current task can be satisfied or not. For Step 7, the initiator of a task requires $O(2mk)$ computations to access enough resources from its residing community. For Steps 9–12, finding a resource-rich community and allocating the current task to that community takes $O(mkq + mk)$ operations. Next, for Steps 13–16, calculating the redundant resources of the accessible communities and accessing the redundant resources from these accessible communities takes $O(3mkq)$ operations. Finally, releasing these less significant tasks one by one until the current task is accomplished (i.e., Steps 17∼21) takes $O(3mnkq)$ complexity. Up to this point, we can determine that allocating each task by the internal Steps 3–24 at most takes $O(mnk + n\log(n) + 3mkq + 3mnkq) = O(3mnkq)$ operations. Because there are n such tasks to allocate, we have the total computational complexity of Algorithm 2 is $O(mn^2kq)$. ■

C. Dependability Analysis

In this section, we will provide a special case when the heuristic algorithm can find the optimal solution.

Theorem 3: For a given *CA-TAP*, if the following three conditions are satisfied.

1) Each agent belongs to at most two communities.
2) For any two adjacent communities C_i and C_j, the sum resources of the nonoverlap agents in C_i and C_j and the overlap agents overlapped between C_i and C_j are sufficient for the tasks residing in them, that is

$$\forall r \in R, \sum_{C\in\{C_i,C_j\}} \sum_{a\in nov-A(C)} rsc(a,r) + \sum_{a\in ov(C_i,C_j)} rsc(a,r) \geq \sum_{t\in T(C_i)\cup T(C_j)} req(t,r).$$

3) For either community C of the two adjacent communities C_i and C_j ($C \in \{C_i, C_j\}$), the sum resources of the nonoverlaps in C and the overlap agents overlapped between C_i and C_j are sufficient for its own community tasks, that is

$$\forall r \in R, \forall C \in \{C_i, C_j\}, \sum_{a\in nov-A(C)} rsc(a,r) + \sum_{a\in ov(C_i,C_j)} rsc(a,r) \geq \sum_{t\in T(C)} req(t,r).$$

Then, we have that all of the tasks can be completed successfully by the heuristic algorithm.

Proof: Without loss of generality, we assume that in the lth round, $T^* = \{t_l^*, t_{l+1}^*, \ldots, t_n^*\}$ is the set of remaining unallocated tasks residing at C_i and C_j and they have been sorted in decreasing order of significance. By using reduction ad absurdum, we suppose that in this round, the first task t_l^* that with the highest significance value cannot be satisfied by the heuristic algorithm. With the earlier discussion, the current task t_l^* can be categorized into nonoverlap and overlap cases.

Nonoverlap case: Task t_l^* is a nonoverlap task, and w.l.o.g., we assume t_l^* belongs to C_i. We denote the tasks that have been completed successfully by the system as $\{t_1^*, t_2^*, \ldots, t_{l-1}^*\}$ and denote the nonoverlap agents' remaining resources of resource type $r (r\in R)$ of community C_i after executing tasks $\{t_1^*, t_2^*, \ldots, t_{l-1}^*\}$ as $\theta(C_i, r)$, the C_j's as $\theta(C_j, r)$, and the overlaps' as $\theta(ov(C_i, C_j), r)$. Obviously, the reason that t_l^*cannot be satisfied is that its certain type resource (e.g., resource type r) cannot be accessed,[2] i.e.

$$\theta(C_i, r) + \theta(ov(C_i, C_j), r) < req(t_l^*, r). \quad (5)$$

By condition 3), the total remaining resources of r of the nonoverlaps in C_i and the overlaps' should be sufficient for t_l^*. Now,the only reason for (5) is that the resources of r of $ov(C_i, C_j)$ have been used for the previous tasks (w.l.o.g., we assume that these tasks belong to C_j). Because of the nonoverlap agent first heuristics, the nonoverlap agents' remaining resources of r in community C_j must be empty, i.e., $\theta(C_j, r) = 0$. Let$\phi(C_i, r), \phi(C_j, r)$,$\phi(ov(C_i, C_j), r)$ be the used resources of resource type r of C_i, C_j and the overlaps $ov(C_i, C_j)$ for tasks$\{t_1^*, t_2^*, \ldots, t_{l-1}^*\}$, respectively. Then, we have

$$\varphi(C_i, r) + \varphi(C_j, r) + \varphi(ov(C_i, C_j), r) = \sum_{t\in\{t_1^*,\ldots,t_{l-1}^*\}} req(t, r). \quad (6)$$

By adding (5) and (6), we can derive

$$\begin{aligned}
&(\theta(C_i, r) + \varphi(C_i, r)) + \varphi(C_j, r) + (\theta(ov(C_i, C_j), r) \\
&\quad + \varphi(ov(C_i, C_j), r)) \\
&= (\theta(C_i, r) + \varphi(C_i, r)) + (\theta(C_j, r) + \varphi(C_j, r)) \\
&\quad + (\theta(ov(C_i, C_j), r) + \varphi(ov(C_i, C_j), r) \\
&= \sum_{C\in\{C_i,C_j\}} \sum_{a\in nov-A(C)} rsc(a, r) + \sum_{a\in ov(C_i,C_j)} rsc(a, r) \\
&< \sum_{t\in\{t_1^*,\ldots,t_{l-1}^*\}} req(t, r) + req(t_i^*, r) \\
&\leq \sum_{t\in T(C_i)\cup T(C_j)} req(t, r).
\end{aligned}$$

Up to this point, we have proved that $\sum_{C\in\{C_i,C_j\}} \sum_{a\in nov-A(C)} rsc(a, r) + \sum_{a\in ov(C_i,C_j)} rsc(a, r) < \sum_{t\in T(C_i)\cup T(C_j)} req(t, r)$ which contradicts condition 2). Therefore, the current task t_l^* can be accomplished by the heuristic algorithm.

Overlap case: The current task t_l^* is an overlap task; the proof is similar to that of nonoverlap case, so we omit the proof for the sake of brevity.

In conclusion, the heuristics can accomplish all of the tasks if the given CA-TAP satisfies the above three conditions. Obviously, this yields the optimal solution. ■

[2]Here, we assume that a task t's accessible agents are either the nonoverlap agents of its residing community or the overlap agents overlapped between C_i and C_j.

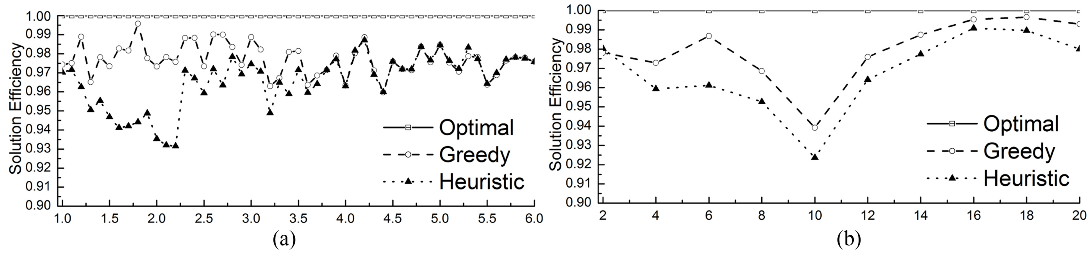

Fig. 4. Solution efficiency comparison of Heuristic, Greedy, and Optimal algorithms (a) on the overlap degree (community number = 6), and (b) on the community number (overlap degree = 1.3).

VI. Experimental Validation and Analyses

In this section, we perform two series of experiments, to accomplish the following:

1) **Validating the advantages of the heuristic algorithm:** we perform two types of tests: a) tests of the performance of the heuristic algorithm on social welfare in the small-scale applications (Sections VI-A1); and b) tests of the scalability of the heuristics to large-scale applications in terms of social welfare and computation time (Section VI-A2).
2) **Validating the advantages of the community-aware task allocation model:** we compare the performance of the community-aware task allocation model with other task allocation models in terms of the social welfare and communication cost (Section VI-B).

A. Validating the Advantages of Heuristic Algorithm

1) *Tests of Performance:*

a) *Network:* To test the advantages of the heuristic algorithm, we apply it to a set of artificial community explicit networks similar to that depicted in Fig. 1. Each network is constructed with 72 nodes divided into several overlapping communities. These nodes are randomly distributed among communities such that each node is a member of overlap ($\geq$1) communities on average (overlap indicates the network overlap degree defined in Definition 2). Next, we connect the nodes, with the probability p_{in} for nodes that belong to the same community and p_{out} for nodes in different communities. The probabilities p_{in} and p_{out} are chosen to keep the networks with explicit community structure (Here, we set $p_{in} = 0.8$ and $p_{out} = 0.2$, which was used in [33]) and to keep the average degree of all nodes equal to a given value.

b) *Experiment setting:* In this experiment, there are 20 tasks submitted to agents randomly and five types of resources available to agents and tasks. The average number of resources required by a task is 30. The payment of a task t is drawn uniformly from the interval $[0, W(t)]$ (which was used in [5]). We set the total number of resources owned by agents is equal to the total number of resources required by tasks and distribute the total available resources to agents uniformly. We compare the performance of the heuristic algorithm with the greedy and the optimal algorithms.

1) **The heuristic algorithm (Heuristic)** is proposed by us. We set the tradeoff parameter α between the task profitability and the task fitness in Definition 7 to 0.8.
2) **Greedy algorithm (Greedy)** [5] arranges tasks in descending order of task profitability first, and then allocates these tasks in turn by a network flow technique.
3) **Optimal algorithm (Optimal)** [3] utilizes an exponential brute-force algorithm to consider all of the relevant combinations of tasks to execute. For each combination, it utilizes the network flow technique to check whether this combination of tasks can be satisfied.

The performance is measured by the solution efficiency (*SE*), which is computed as follows:

Definition 9: Solution Efficiency. If there are two algorithms X and Y for a *CA-TAP*, and their social welfares are $SW(X)$ and $SW(Y)$, respectively. Then, the solution efficiency of X relative to Y is defined as $SE(X/Y) = SW(X)/SW(Y)$.

In this test, we compare the solution efficiencies of the heuristic algorithm and the greedy algorithm relative to the optimal algorithm, i.e., **Heuristic** = SE(**Heuristic/Optimal**), **Greedy** = SE(**Greedy/Optimal**).

c) *Simulation results:* Fig. 4 shows the solution efficiencies of the greedy and the heuristic algorithms on the overlap degree (Fig. 4(a)) and on the community number (Fig. 4(b)). From the experimental results in Fig. 4, we conclude the following:

1) In Fig. 4(a): The solution efficiency of the Heuristic drops from 0.97 to 0.93 as the overlap degree varies from 1.0 to 2.2 and revives when the overlap degree varies in the range [2.3, 6.0], and can eventually reach 0.98. This trend can be explained by the fact that when there are no overlap agents (i.e., *overlap* = 1.0), the Heuristic can allocate tasks to agents without any collision, which performs as well as the optimal algorithm. Once there are some overlaps (e.g., *overlap* = 1.1∼2.2), the optimal can lead the overlap agents to execute the more desirable tasks, while the overlaps of the Heuristic perform not as well. However, if the overlap degree becomes larger to some extent (e.g., *overlap* = 2.3∼6.0), agents can negotiate with a number of community partners and then nearly all of the tasks can be completed even by the Heuristic. Therefore, the Heuristic can obtain a relatively good performance in these cases with larger overlap degree.
2) In Fig. 4(b): The solution efficiency of the Heuristic drops from 0.98 to 0.92 as the community number varies from 2 to 10 and rebounds when the network is highly separated (i.e., community number = 12∼20).

This trend can be explained by the fact that when the network is partitioned into large size communities (e.g., community number = 2), the agents of the Heuristic can find sufficient resources for tasks as well as finding the optimum. In case that the network is highly separated (e.g., community number = 20), only a few tasks can be successfully completed even by the optimal approach due to the limited number of community partners.

3) In all of the experiments, Greedy performs very close to the Optimal (*SE*(**Greedy/Optimal**)>0.94) due to its high computational complexity (the detailed comparison of the Heuristic and the Greedy is shown in Section VI-A2). This finding verifies the rationale that we compare the Heuristic with the Greedy in large-scale applications on social welfare.

In conclusion, the heuristic algorithm is effective compared with the optimal solution in terms of the social welfare: in the worst case (*overlap* = 1.3 and community number = 10), the solution efficiency is still higher than 0.92.

2) *Tests of Scalability:*

a) *Dataset and experiment setting: Dataset:* To study the scalability of the heuristics to real large-scale applications, the dataset we utilize in this experiment is the scientist co-authorship network [34], in which there are 1589 scientists from a broad variety of fields, and an edge between authors *i* and *j* is included in the network if they co-authored a paper. We utilize the overlapping community detection algorithm [33] to divide the network into 300 overlapping communities with the overlap degree be equal to 1.034. *Experiment setting:* We vary the number of tasks from 100 to 500 with steps of 100. The other settings are similar to those described in Section VI-A1.

Because it is not feasible to compute the optimal solution in large-scale applications, and in Section IV-A1, we have verified that the Greedy is very close to the optimal solution (*SE*(**Greedy/Optimal**)>0.94) in various scenarios. Thus, it makes sense to validate the scalability of the Heuristic to large-scale applications by comparing it with the Greedy on solution efficiency and computation time.

b) *Simulation results:* Table II shows the scalability results of the heuristic algorithm. Through the experimental results in Table II, we determine the following:

1) The heuristic algorithm performs very close to the greedy algorithm on social welfare for any large-scale cases (the average solution efficiency approximates 0.99).

2) The computational load of the Heuristic is significantly reduced compared with that of the Greedy (e.g., the runtime of the Greedy is 1.16×10^5 times that of the Heuristic when the task number reaches 500). Before explaining the phenomenon, it is necessary to introduce the network flow technique that is involved in the greedy and optimal algorithms briefly. In [5], a flow network is constructed as follows: 1) create a source node *s* and a sink node *s'*; 2) for each agent a_i and each resource type r_k, if $rsc(a_i,r_k)$>0, then create an agent resource node $a_{i,rk}$ and an edge from the source node *s* to this node with capacity $rsc\ (a_i,r_k)$; 3) for each task t_j and each resource type r_k, if $req(t_j,\ r_k)$>0, then create a task resource node $t_{j,rk}$ and an edge from this node to the sink node *s'* with capacity $req(t_j,r_k)$; 4) for each agent a_i and each resource type r_k, connect the agent resource nodes $a_{i,rk}$ to its accessible task resource nodes $t_{j,rk}$ (i.e., $\delta(a_i,int(t_j)) = 1$), and give this connection unlimited capacity; and 5) solve the maximum flow problem and check whether the maximum flow is sufficient for the resources required by the set of allocated tasks or not. In this paper, we utilize the Basic Ford–Fulkerson algorithm [35] to solve the maximum flow problem. Thus, to check the total *n* tasks, the greedy takes $O(n(x_a+y_t)(qx_a y_t/mk)^2)$ operations to return the allocation result, where m, k, q indicate the numbers of agents, resource types, and communities in a *CA-TAP*, and x_a, y_t, x_a+y_t, and $qx_a y_t/mk$ indicate the numbers of agent resource nodes, task resource nodes, total nodes and total edges in the constructed flow network of the *CA-TAP*.

TABLE II

PERFORMANCE IN LARGE-SCALE NETWORK

Tasks	Run time(s)		$SE(\frac{\text{Heuristic}}{\text{Greedy}})$
	Greedy	Heuristic	
100	3.4	1.5×10^{-3}	0.9993
200	110	5.1×10^{-3}	0.9990
300	808	1.5×10^{-2}	0.9951
400	2700	3.6×10^{-2}	0.9847
500	7642	6.6×10^{-2}	0.9953

Now, we are ready to explain the phenomenon that the heuristic algorithm is superior to the greedy algorithm on the running time. The running time of the greedy algorithm mainly depends on the numbers of agent resource nodes and task resource nodes in the constructed flow network. In the worst case, the numbers of agent resource nodes and task resource nodes of a flow network are *mk* and *nk*, respectively (*m* and *n* indicate the numbers of agents and tasks), whose complexity is $O(n(mk+nk)(qnk)^2)$. As discussed in Section V-B, the complexity of the heuristic algorithm is $O(3mn^2kq)$. The time complexity ratio between greedy algorithm and heuristic algorithm then is $O(n(mk+nk)(nqk)^2)/O(3mn^2kq) = (m+n)nqk^2/(3m)$. (It approximates 1.74×10^6 when there are 500 tasks, which is in accordance with the experimental results).

In conclusion, in large-scale scenarios, the heuristic algorithm achieves approximately the same social welfare as the greedy algorithm, but its computational load is significantly reduced. In fact, for large-scale applications, the greedy algorithm will be computationally infeasible. Thus, the heuristic algorithm is probably a better choice to achieve a relatively efficient social welfare with limited computational cost.

B. Validating the Advantages of Community-Aware Task Allocation Model

a) *Network and Experiment Setting:* In this section, we compare the performance of task allocation models on

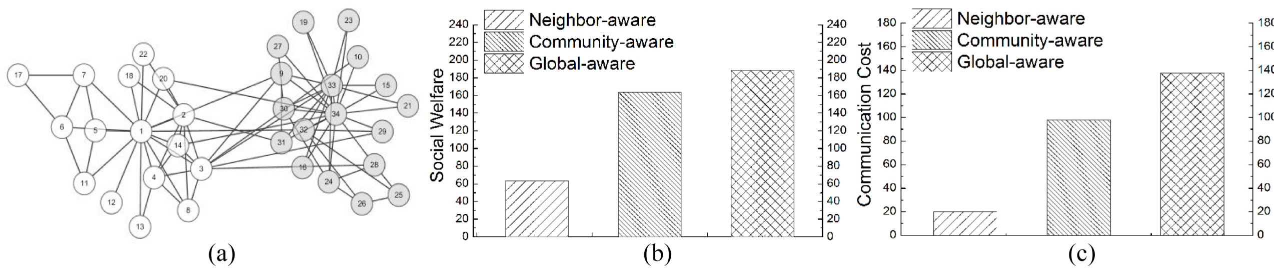

Fig. 5. Performance comparison of global-aware, community-aware, and local neighbor-aware task allocation models on the real-world applications in terms of (b) social welfare and (c) communication cost. (a) is the real-world karate club friendship network studied by Zachary [17], the nodes belong to the common community are with the same color.

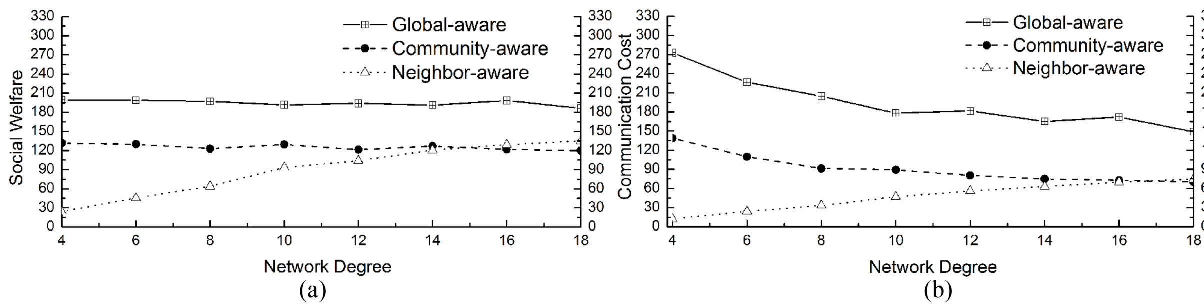

Fig. 6. Performance comparison of global-aware, community-aware, and local neighbor-aware task allocation models on the network degree in terms of (a) social welfare and (b) communication cost. Here, we fix the overlap degree to 1.3.

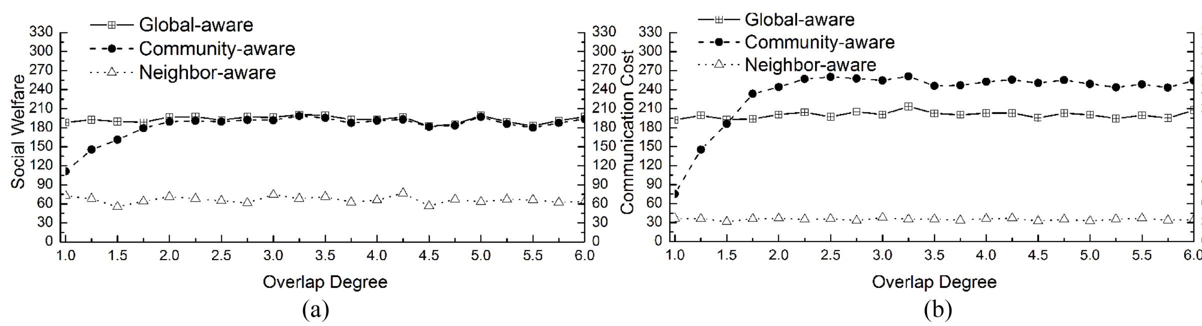

Fig. 7. Performance comparison of global-aware, community-aware, and local neighbor-aware task allocation models on the overlap degree in terms of (a) social welfare and (b) communication cost. Here, we fix the network degree to 8.

a real-world friendship network and the computer generated networks. The real-world friendship network is The Zachary's karate club [17], in which there are 34 club members and 78 edges among these members. An edge between members i and j is included in the network if they are friends. Moreover, this karate club is constituted of two communities which are already known in reality (see Fig. 5(a)). The computer generated networks and the experiment settings are similar to those described in Section VI-A1. We compare the community-aware task allocation model with the local neighbor-aware and the global-aware task allocation models.

1) **The community-aware task allocation model (Community-aware)** is proposed by us. In this model, agents can cooperate only with its intracommunity partners.
2) **The local neighbor-aware task allocation model (Neighbor-aware)** such as[5] constrains the cooperation within immediate neighbors. In this model, a central controller utilizes the network flow technique to allocate tasks in decreasing order of their profitability.
3) **The global-aware task allocation model (Global-aware)**, such as [13] permits agent to cooperate with any other system agents. This model sorts all system tasks in decreasing order of profitability first, and then allocates them by a breadth first negotiation approach.

The performance of these task allocation models is evaluated by social welfare and communication cost. The social welfare is the sum of the payments of these tasks that are completed successfully. The communication cost is computed as follows: if an initiator agent a_i negotiates with its community partners a_1, a_2 and a_3 for its initiated tasks t_j, the communication cost of agent a_i for task t_j is: $Ccost(a_i,t_j)=dist(a_i,a_1)+dist(a_i,a_2)+dist(a_i, a_3)$, where $dist(a_x,a_y)$ is the length of the shortest path between agents a_x and a_y.

b) *Simulation results:* Fig. 5 shows the performance comparison of the community-aware (Community), local neighbor-aware (Neighbor), and global-aware (Global) task allocation models in the real-world applications. From the experimental results in Fig. 5, we can determine that the social welfare of the Community performs very close to that of the

Global, which far exceeds that of the Neighbor (Fig. 5(b)). This can be explained by the fact that in the neighbor-aware model, each member has average four partners to cooperate with; while in the community-aware model, the members in white community has 16 partners to cooperate with and these members in gray community has 18 partners. Obviously, there will be more tasks completed successfully in the community-aware model than the local neighbor-aware model. On the other hand, the Community reduces the communication load to a large extent compared with the Global (Fig. 5(c)).

Fig. 6 shows the performance comparison of the Community, Neighbor and Global task allocation models on the network degree. From the experimental results in Fig. 6, we conclude the following:

1) In Fig. 6(a), when the network degree varies from 4 to 18, the social welfares of the Global and the Community stay almost invariant, while the Neighbor's is in direct proportion to it. This can be explained by the fact that in the Neighbor model, the more neighbors, the more resources that agents can access, and the more tasks will be completed successfully. In contrast, in the Community model, the intracommunity agents are always allowed to cooperate with each other regardless of whether there exists a connection between a pair of community partners or not. It should also be noticed that when the network degree becomes large enough (e.g., network degree$\geq$16), the social welfare of Community is even smaller than the Neighbor's. The reason is that when the network degree$\geq$16, the number of tasks that be accomplished in the Neighbor model is larger than the number of tasks accomplished in the Community model.
2) In Fig. 6(b), the communication cost of the Neighbor is in direct proportion to the network degree, while the communication costs of the Global and Community are in inverse proportion to the network degree. The potential reason is that when the network degree becomes larger, agents negotiate with each other easily (i.e., the average social distance among agents becomes shorter) for the Community and Global models. Notice also that when the network degree reaches 18, the communication cost incurred by the Community is even smaller than that of the Neighbor, which is much less than that of the Global.

Fig. 7 shows the performance comparison of the Community, Neighbor, and Global models on the overlap degree. From the experimental results in Fig. 7, we have the following observations:

1) In Fig. 7(a), as the overlap degree increases, the social welfare of the Community performs better as well, while the social welfares of the Global and Neighbor stay almost the same. The potential reason is that in the Community model, the more overlaps, the more community partners that agents can cooperate with and then the more tasks will be completed successfully. For the case *overlap*$\geq$2.0, the social welfare yielded by Community performs approximately as large as the Global, which far exceeds that of the Neighbor.
2) In Fig. 7(b), when *overlap* ranges from 1.0 to 2.0, the communication cost of the Community is in direct proportion to it and when overlap degree becomes larger to some extent (i.e., *overlap*>2.0), the communication cost of the Community remains almost flat. The potential reason is that in the latter cases (i.e., *overlap*>2.0), the Community model has reached its extreme capacity for accomplishing tasks (this can also be inferred from Fig. 7(a)). Moreover, when overlap varies from 1.0 to 1.5, the communication cost of the Community is smaller than that of the Global, while *overlap*>1.5, the communication cost of the Community becomes larger than that of the Global. This can be explained by the fact that in our heuristic algorithm of the Community model, the initiator agent first negotiates with the nonoverlap agents and then negotiates with the overlap agents. However, in the Global model, the initiator agent negotiates with agents from nearby to faraway gradually. In case that the nonoverlap agents have longer negotiation distance, the Community will produce more communication cost than the Global.

In conclusion, on one hand, the community-aware task allocation model increases the social welfare greatly compared with the neighbor-aware model. On the other hand, it reduces the communication cost to a large extent compared with the global-aware model. Thus, the community-aware task allocation model is a favor option to achieve a relatively higher system overall profit with less communications cost.

VII. Conclusion

In this paper, we introduce a new variant of task allocation model for SN-MASs, where agent negotiates only with its intracommunity members. Under such community-aware scenarios, we prove that it is NP-hard to maximize the social welfare and to minimize system communication cost. To solve such an NP-hard problem effectively, we introduce a heuristic algorithm in which tasks are first arranged in a decreasing order of significance and then a significant task-first, nonoverlap agent-first, and breadth-first heuristics is utilized to allocate the sorted tasks in turn. We also conduct a series of experiments to validate the advantages of this community-aware task allocation model. From the experiments, we can find that: 1) in our community-aware model, besides these direct neighbor partners, agents can also cooperate with a number of other indirect community members, which will yield more system overall profit compared to the local neighbor-aware model; 2) in our community-aware model, because of the dense intracommunity connections, it is easy for the community members to cooperate, which will produce less system communication cost compared to the global-aware task allocation model; and 3) because of the lower time complexity of the proposed heuristic algorithm, our community model can be exploited well in large-scale applications.

One limitation of this paper is that we assume agents are cooperative; in other words, agents always contribute their idle resources to desirable tasks from the system perspective. However, this assumption may not be realistic, for example, in the economics, when an agent is asked to perform tasks, it always serve for the optimal task that maximizes its own benefit rather than maximizes system overall profit. In our future work, we will investigate the effect of agent selfish behavior on the community-aware task allocation problem and extend our centralized algorithm to a totally distributed fashion, which can be applied to the situation where agents are selfishness.

Another interesting topic for the future work is to consider the dynamic community structure in SN-MASs. In this paper, the communities are fixed during task allocation. However, in reality the communities may be dynamic [8], [36]. Such a dynamic situation may bring about new problems to our current task allocation model, for example, due to agents join and leave communities dynamically, the task execution may be unsuccessful. Therefore, it is essential to devise feasible approaches to deal with the emergent problems in dynamic SN-MASs.

Acknowledgments

The authors would like to thank the editors and reviewers for their useful comments and suggestions.

References

[1] R. Guimerà, S. Mossa, A. Turtschi, and L. A. N. Amaral, "The worldwide air transportation network: Anomalous centrality, community structure, and cities' global roles," *Proc. Nat. Acad. Sci.*, vol. 102, no. 22, pp. 7794–7799, 2005.

[2] W. Wang and Y. Jiang, "Migration cost-sensitive load balancing for social networked multiagent systems with communities," in *Proc. IEEE Int. Conf. Tools With Artif. Intell.*, Nov. 2013, pp. 127–134.

[3] M. de. Weerdt, Y. Zhang, and T. Klos, "Distributed task allocation in social networks," in *Proc. 6th AAMAS*, May 2007, pp. 500–507.

[4] S. Lozano, A. Arenasand, and A. Sanchez, "Mesoscopic structure conditions the emergence of cooperation on social networks," *PLoS ONE* vol. 3, no. 4, p. e1892, 2008.

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

[6] S. Brânzei and K. Larson, "Social distance games," in *Proc. 22nd IJCAI*, vol. 1. Jul. 2011, pp. 91–96.

[7] L. Backstrom, D. Huttenlocher, J. Kleinberge, and X. Y. Lan, "Group formation in large social networks: Membership, growth, and evolution," in *Proc. 12th SIGKDD*, Aug. 2006, pp. 44–54.

[8] C. Tantipathananandh, T. B. Wolf, and D. Kempe, "A framework for community identification in dynamic social networks," in *Proc. 13th SIGKDD*, Aug. 2007, pp. 717–726.

[9] W. Chen, Z. Liu, X. Sun, and Y. Wang, "Community detection in social networks through community formation games," in *Proc. 22nd. IJCAI*, vol. 3. Jul. 2011, pp. 2576–2581.

[10] B. An, S. K. Mong, L. G. Tang, S. Q. Li, and D. J. Cheng, "Continuous-time negotiation mechanism for software agents," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 36, no. 6, pp. 1261–1272, Dec. 2006.

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

[12] R. Kota, N. Gibbins, and N. R. Jennings, "Self-organising agent organisations," in *Proc. 8th AAMAS*, May. 2009, pp. 797–804.

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

[14] Y. Jiang and Z. Li, "Locality-sensitive task allocation and load balancing in networked multiagent systems: Talent versus centrality," *J. Parallel Distrib. Comput.*, vol. 71, no. 6, pp. 822–836, 2011.

[15] Y. Jiang and Z. Huang, "The rich get richer: Preferential attachment in the task allocation ofcooperative networked multiagent systems with resource caching," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 42, no. 5, pp. 1040–1052, Sep. 2012.

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

[17] W. W. Zachary, "An information flow model for conflict and fission in small groups," *J. Anthropological Res.* vol. 33, no. 4, pp. 452–473, 1977.

[18] M. Boss, H. Elsinger, M. Summer, and S. Thurner, "Network topology of the interbank market," *Quant. Fin.* vol. 4, no. 6, pp. 677–684, 2004.

[19] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *Proc. Nat. Acad. Sci. United States Amer.*, vol. 99, no. 12, pp. 7821–7826, 2002.

[20] G. Palla, I. Derenyi, I. Farkas, and T. Vicsek, "Uncovering the overlapping community structure of complex networks in nature and society," *Nature*, vol. 435, no. 7043, pp. 814–818, 2005.

[21] E. Jaho, M. Karaliopoulos, and I. Stavrakakis, "Social similarity favors cooperation: The distributed content replication case," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 3, pp. 601–613, Mar. 2013.

[22] S. M. Allen, G. Colombo, and R. M. Whitaker, "Cooperation through self-similar social networks," *ACM Trans. Auto. Adaptive Syst.*, vol. 5, no. 1, pp. 1–29, 2010.

[23] J. H. W. Tan and D. J. Zizzo, "Groups, cooperation and conflict in games," *J. Socio-Econ.*, vol. 37, no. 1, pp. 1–17, 2008.

[24] S. Kraus, O. Shehory, and G. Taase, "Coalition formation with uncertain heterogeneous information," in *Proc. 2nd AAMAS*, Jul. 2003, pp. 1–8.

[25] E. Manisterski, E. David, S. Kraus, and N. R. Jennings, "Forming efficient agent groups for completing complex tasks," in *Proc. 5th AAMAS*, May 2006, pp. 834–841.

[26] O. Shehory and S. Kraus, "Methods for task allocation via agent coalition formation," *Artif. Intell.*, vol. 101, nos. 1–2, pp. 165–200, 1998.

[27] T. Michalak, J. Sroka, T. Rahwan, M. Wooldridge, P. McBurney, and N. R. Jennings, "A distributed algorithm for anytime coalition structure generation," in *Proc. 9th AAMAS*, May 2010, pp. 1007–1014.

[28] S. Abdallah and V. Lesser, "Learning the task allocation game," in *Proc. 5th AAMAS*, May 2006, pp. 850–857.

[29] D. Lai, C. Nardini, and H. Lu, "Partitioning networks into communities by message passing," *Phys. Rev. E*, vol. 83, no. 1, p. 016115, 2011.

[30] O. Kullmann, "New methods for 3-SAT decision and worst-case analysis," *Theoret. Comput. Sci.*, vol. 223, nos. 1–2, pp. 1–72, 1999.

[31] D. S. Johnson, M. Yannakakis, and C. H. Papadimitriou, "On generating all maximal independent sets," *Inf. Process. Lett.*, vol. 27, no. 3, pp. 119–123, 1988.

[32] R. Grant, A. Jammine, and H. Thoms, "Diversity, diversification, and profitability among british manufacturing companies, 1972–84," *Acad. Manag. J.*, vol. 31, no. 4, pp. 771–801, 1988.

[33] S. Gregory, "An algorithm to find overlapping community structure in networks," *Lecture Notes Artif. Intell.*, vol. 4702, pp. 91–102, Sep. 2007.

[34] M. E. J. Newman, "Finding community structure in networks using the eigenvectors of matrices," *Phys. Rev. E*, vol. 74, no. 3, p. 036104, 2006.

[35] L. R. Ford and D. R. Fulkerson, "Maximal flow through a network," *Canadian J. Math.*, vol. 8, no. 3, pp. 399–404, 1956.

[36] D. Greene, D. Doyle, and P. Cunningham, "Tracking the evolution of communities in dynamic social networks," in *Proc. ASONAM*, Aug. 2010, pp. 176–183.

Wanyuan Wang (S'13) received the B.S. degree in information and computing science from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2011. He is currently pursuing the Ph.D. degree at the Distributed Intelligence and Social Computing (DISC) Laboratory, School of Computer Science and Engineering, Southeast University, Nanjing, China.

His current research interests include social networks and multiagent systems.

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 (DISC) Laboratory, School of Computer Science and Engineering, Southeast University, Nanjing, China. He has published over 70 scientific articles in refereed journals and conference proceedings, such as the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS-PART A: SYSTEMS AND HUMANS, the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS-PART C: APPLICATIONS AND REVIEWS, the *Journal of Parallel and Distributed Computing*, the International Joint Conference on Artificial Intelligence (IJCAI), the International Conference on Autonomous Agents and Multiagent Systems (AAMAS), and the IEEE International Conference on Tools with Artificial Intelligence (ICTAI). His current research interests include multiagent systems, social networks, social computing, and distributed intelligent systems.

Dr. Jiang is 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 the *Operations Research and Fuzziology*, and a member of the editorial board of the *Chinese Journal of Computers*.