A

Reliable Task Allocation with Load Balancing in *Multiplex* Networks

YICHUAN JIANG, YIFENG ZHOU and YUNPENG LI, Southeast University

In multiplex networks, agents are connected by multiple types of links; a multiplex network can be split into more than one network layer that is composed of the same type of links and involved agents. Each network link type has a bias for communicating different types of resources; thus, the task's access to the required resources in multiplex networks is strongly related to the network link types. However, traditional task allocation and load balancing methods only considered the situations of agents themselves and did not address the effects of network link types in multiplex networks. To solve such problem, this paper considers both link types and agents and substantially extends the existing work by highlighting the effect of network layers on the task allocation and load balancing; two multiplex network-adapted models of task allocation with load balancing are then presented, which are network layer-oriented allocation and agent-oriented allocation. Moreover, this paper addresses the unreliability in multiplex networks, which includes the unreliable links and agents, and implements a reliable task allocation based on a negotiation reputation and reward mechanism. Our findings show that both of our presented models can effectively and robustly satisfy the task allocation objectives in unreliable multiplex networks; the experiments prove that they can significantly reduce the time costs and improve the success rate of tasks for multiplex networks over the traditional simplex network-adapted task allocation model. Lastly, we find that our presented network layer-oriented allocation performs much better in terms of reliability and allocation time compared to our presented agent-oriented allocation, which further explains the importance of network layers in multiplex networks.

Categories and Subject Descriptors: I.2.11 [**Computing Methodologies**]: Distributed Artificial Intelligence–*Multiagent systems*

General Terms: Design, Algorithms, Performance, Experimentation

Additional Key Words and Phrases: Multiplex networks, multiagent systems, social networks, task allocation, load balancing, reliability, network layers

ACM Reference Format:
Yichuan Jiang, Yifeng Zhou, and Yunpeng Li, 2014. Reliable Task Allocation with Load Balancing in Multiplex Networks. *ACM Trans. Autonom. Adapt. Syst.* V, N, Article A (January YYYY), 33 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

Some preliminary results of the network layer-oriented allocation model in this paper were presented in Proceedings of the 25th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'13) [Jiang et al. 2013a].
This work is supported by the National Science Foundation of China(No.61170164 and No.61472079), the Program for Distinguished Talents of Six Domains in Jiangsu Province (No.2011-DZ023), and the Funds for Distinguished Young Scholars of Jiangsu Province (BK2012020).
Author's addresses: The authors are with the School of Computer Science and Engineering, Southeast University, Nanjing 211189, China, and also with the Key Laboratory of Computer Network and Information Integration (Southeast University), Ministry of Education, Nanjing 211189, China. E-mail: yjiang@seu.edu.cn; njtujyc@hotmail.com.

DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

The interest in developing networked distributed systems has increased over the past decade [Liu et al. 2005; Abdallah and Lesser 2007; Das and Islam 2012]. In fact, most network applications, such as social networks, transportation networks, and P2P networks, can be viewed as networked multiagent systems (NMASs), in which agents represent the autonomous nodes and interaction relations represent the interconnections among nodes [Das and Islam 2012; Jiang and Huang 2012; Ye et al. 2013]. The concept of NMASs enables users to represent networked distributed systems at an abstract level without addressing the particulars of the target systems [Abdallah and Lesser 2007]. When a NMAS intends to execute tasks, these tasks will be allocated to some agents according to a certain criterion, which is called task allocation [Shehory and Kraus 1998; Kraus and Plotkin 2000]; if too many tasks are crowded on certain agents, tasks could be switched from heavy-burdened agents to light-burdened ones to reduce the waiting time of tasks at agents, which is called load balancing [Liu et al. 2005].

In NMASs, some resources are placed within the networks and can be accessed by agents to execute tasks. Therefore, many existing models of task allocation and load balancing have been implemented based on the accessibility of required resources [An et al. 2011; Weerdt et al. 2012; Jiang and Jiang 2009; Jiang et al. 2013b], i.e., an agent's probability of being allocated tasks is determined by its accessibility to required resources for the tasks. *Most existing studies on this subject are based on the assumption that all the links in the network are of the same type, i.e., the underlying network is assumed to be simplex.*

However, in reality, multiplex networks are often seen where there are multiple types of links between agents [Gómez-Gardeñes et al. 2012; Yağan and Gligor 2012; Szell et al. 2010; Brummitt et al. 2012]. In multiplex networks, each type of links may have different relative biases in communicating different types of resources [Yağan and Gligor 2012]. For example, in a multiplex transportation network, an express letter may prefer air transportation, but bulk goods may prefer railway transportation; in a multiplex social network, the friendship links easily communicate entertainment resources, but scientific collaboration links easily communicate academic resources [Gómez-Gardeñes et al. 2012]. Therefore, *an agent's accessibility to a resource is determined not only by the localities of the agent and the resource in the network but also by the link types between them.*

To address the above situation in multiplex networks, this paper extends the existing resource based task allocation and load balancing approaches as follows: 1) the resource negotiation between agents considers both the agent localities in networks and the link types with varying communication biases for different resources; 2) the accessibility of an agent for resources is measured by both the communication distances and the link types; and 3) the targets of task allocation and load balancing include not only agents but also network layers.

Moreover, agents are often attributed to different organizations due to the multiplicity of the agents' links [Szell et al. 2010]. Therefore, some agents may not provide reliable resources for other agents in different network layers, and some types of links may also be unreliable for communicating certain types of resources. Therefore, the problem of unreliable resources [Jiang et al. 2013b; Ohtsuki et al. 2006] needs to be solved in the multiplex networks.

Thus, this paper's main contribution is to present systematic research on reliable task allocation with load balancing in multiplex networks, which extends previous benchmark works by introducing the factor of multiple link types into task allocation for the first time. The rest of this paper is organized as follows. In Section 2, we

compare our work with the related work on the subject; in Section 3, we present the problem description; in Section 4, we model the multiplex networks; in Section 5, we propose the model of reliable task allocation with load balancing; in Section 6, we provide the experimental results; finally, we discuss and conclude our paper in Section 7.

2. RELATED WORK

To the best of our knowledge, systematic previous works on task allocation and load balancing for multiplex networks have not yet been presented. We now introduce the related work of task allocation and load balancing in simplex networks and unreliable systems; then, we introduce the related work of multiplex networks.

2.1. Task Allocation and Load Balancing in Simplex Networks

Existing related studies in simplex networks can be categorized as follows: 1) self-owned resource-based methods vs. contextual resource-based methods (according to the resource negotiation mechanisms for tasks); 2) centralized control methods vs. distributed control methods (according to the allocation control mechanisms); and 3) passive adaptation vs. proactive adjustment for networks (according to the adjustment mechanism for the constraints of network structures).

2.1.1. Self-Owned Resource-Based Methods vs. Contextual Resource-Based Methods. Many related works have aimed to optimize the execution time of tasks. Task execution in NMASs can be described via the agents' operations when accessing required resources in the networks [Liu et al. 2005; An et al. 2011; Weerdt et al. 2012; Jiang et al. 2013b]; therefore, resource accessibility can significantly influence the task's execution time. Many related works have been implemented based on two types of resources: 1) *a self-owned resource-based approach* implemented based on the agents' self-owned resources status, i.e. an agent's probability of being allocated tasks is determined only by its self-owned resources [Liu et al. 2005; Kraus and Plotkin 2000]; and 2) *a contextual resource-based approach* implemented based on not only the agent' self-owned resources status but also their contextual resources status because agents may cooperate with others within their contexts when they execute tasks [Jiang and Huang 2012; Jiang and Jiang 2009].

Generally, previous works have assumed that the underlying networks are simplex. In comparison, our study considers the characteristics of multiple links in the communicating paths for resources and implements the task allocation and load balancing based not only on the situations of agents' resources but also on the situations of network layers' resources.

2.1.2. Centralized Control Methods vs. Distributed Control Methods. Related works can also be categorized into centralized and distributed approaches according to their allocation control mechanisms [Zhang et al. 2012]. In the centralized approach, a central controller needs to know the information of the whole system to implement task allocation [Fjuita and Lesser 1996]. In the distributed approach, a central controller is not necessary; a typical distributed method is the contract net protocol [Smith 1980], a wellknown task sharing protocol in which each agent in a network can be a manager or a contractor at different times or for different tasks.

In our previous work [Jiang and Jiang 2009], we provided a spectrum between a totally centralized approach and a totally decentralized approach to task allocation: the manager is selected by the centralized heuristic that can be utilized to control the overall status information, and the contractors are selected by the distributed heuristic that can be utilized to achieve the flexibility of task allocation in large NMASs. In our previous study [Jiang and Jiang 2009], the manager is fixed throughout the whole

allocation process for one task; in comparison, this paper proposes a method where the manager can be altered during the allocation process according to the current intermediate allocation results. This mechanism allows the manager to be adaptable to current network structures. Moreover, both agents and network layers can act as the manager/contractor in this paper.

2.1.3. Passive Adaptation vs. Proactive Adjustment for Network Structures. To consider the effects of network structures in task allocation and load balancing, the related work can be categorized into two types.

The first type is a passive adaptation approach, in which task allocation is implemented by *passively satisfying* the constraints of the network structure. For example, Weerdt et al. [2012] developed an algorithm based on the contract-net protocol to implement distributed task allocation for social networks.

Another type is a proactive adjustment approach, in which task allocation is implemented by *proactively adjusting* network structures to improve the performance. Kota et al. [2012] presented a decentralized approach to structural adaptation by explicitly modeling problem-solving agent organizations. Their approach enables agents to modify their structural relationships to improve the allocation of tasks; and agents can set the edge weights to either 0 (disconnected) or 1 (connected) for task allocation.

Overall, the works described above did not consider the multiplicity of links in the network. In comparison, our study satisfies the multiplex characteristics of links in multiplex networks.

2.2. Task Allocation and Load Balancing in Unreliable Systems

Various studies have addressed unreliable systems to a certain extent using game theory and mechanism design. For example, a representative work by Weerdt et al. [2012] presented a mechanism design approach that can incentivize self-interested agents to correctly report their private information; Zlotkin and Rosenschein [1991] presented a negotiation mechanism that can address incomplete information and deception in unreliable systems; Shehory and Kraus [1998] presented algorithms for task allocation among agents via coalition formation, which can motivate the agents to act in order to maximize the benefits of the system as a whole so that the deceptive problem can also be addressed to a certain extent. However, these approaches did not consider the objective of minimizing the access time of resources in networks, which is crucial to the performance of task execution in NMASs. Furthermore, they also did not consider the multiple link characteristics of multiplex networks.

In our previous work [Jiang et al. 2013b], we presented a task allocation model for unreliable multiagent systems in simplex social networks, which can achieve reliable resources with the least access time to execute tasks. However, the work did not consider the multiplicity of links in the network. In comparison, this paper substantially extends the model in our previous study [Jiang et al. 2013b] by considering the characteristics of links in multiplex networks and implementing reliable task allocation not only to agents but also to network layers.

2.3. Multiplex Networks

Many real-world systems can be modelled as a set of interdependent networks or networks with multiple types of connection links, which are called multiplex networks [Salehi et al. 2014; Lee et al. 2012]. In multiplex networks, agents are connected by multiple types of links [Yağan and Gligor 2012]; for example, people in a multiplex social network interact via their friendships, family, or colleague relationships. In multiplex networks, each type of links may have different relative biases in communicating different types of resources [Yağan and Gligor 2012]; for example, in a multiplex trans-

portation network, an express letter may prefer air transportation, but bulk goods may prefer railway transportation. Therefore, the communication in multiplex networks should consider the factors of link characteristics and network layers.

Generally, the existing studies of multiplex networks mainly include the following three aspects: modeling the structures of multiplex networks, diffusion processes in multiplex networks, and cooperation in multiplex networks [Boccaletti et al. 2014]

2.3.1. Modeling the Structures of Multiplex Networks. In related work, how to model the structural characteristics of multiplex networks is a crucial problem [Jiang and Jiang 2014]. For example, Gómez et al. [2013] modeled the multiplex networks as structured multi-level graphs in which the interconnections between the layers determine how a given agent in a layer and its counterpart in another layer are linked and influence each other; Buldyrev et al. [2010] presented a method for modeling multiplex networks that concerned the coupling and interdependence among different networks. In this paper, we mainly model the structures of multiplex networks based on our previous conference paper [Jiang et al. 2013a] that modeled a multiplex network as a set of associative network layers; in our model, each network layer is composed of links of the same type and the involved agents and is associated with different communication biases for different types of resources.

2.3.2. Diffusion Processes in Multiplex Network. Recently, there are a non-trivial amount of related studies that concerned about the diffusion processes in multiplex networks [Salehi et al. 2014]. In related studies on the diffusion in multiplex networks, a common method is extending the existing diffusion models in simplex networks by considering the multiplicity of links [Li and Jiang 2014]. Two typical classes of approaches are: generalizing the epidemic model and the threshold model in simplex networks to the multiplex networks. For example, Cozzo et al. [2013] extended the traditional epidemic model in simplex networks, SIS (susceptible-infected-susceptible) model, and proposed a contact-based Markov chain approach to study epidemic-like social contagion in multiplex networks; Brummitt et al. [2012] studied cascades in multiplex social networks by generalizing the threshold diffusion model, in which an agent activates if a sufficiently large fraction of its neighbors in any type of link are active.

2.3.3. Cooperation in Multiplex Networks. Cooperation in multiplex networks is another issue in existing studies, where game theory is often used. It is generally accepted that interdependence between network layers in multiplex networks does promote cooperation by means of organizational complexity and enhanced reciprocity that is out of reach on isolated networks; then, Wang et al. [2013] determined how strong the interdependence between the network layers really ought to be for the optimal promotion of cooperation. Gómez-Gardeñes et al. [2012] showed that multiplexity enhances the resilience of cooperation to defection, which relies on a nontrivial organization of cooperative behavior across network layers. On the other hand, Wang et al. [2014] found that degree mixing in two-layer scale-free networks impedes the evolution of cooperation.

Overall, the existing studies on multiplex networks mainly considered the aspects of structural modelling, diffusion processes, and cooperation. However, they did not address the issues of task allocation and load balancing. Therefore, this paper investigates the task allocation and load balancing in multiplex networks for the first time.

3. PROBLEM DESCRIPTION

3.1. Task Allocation with Load Balancing in NMASs

3.1.1. Formal Definitions

Definition 3.1 (*Task Allocation in NMASs*). [Jiang et al. 2013b]. Given a NMAS, $N =< A, E >$, where A is the set of agents and each agent owns a different set of resources, and $\forall a_i, a_j \in E$ indicates the existence of a network link between agent a_i and a_j, the set of resources in agent a_i is assumed to be R_{ai}, and the set of resources required by task t is assumed to be R_t. If task t arrives at the network, the task allocation in NMAS can be defined as the mapping of task t to a set of agents, A_t, which can satisfy the following situations:

1) The resource requirements of t can be satisfied, i.e.,$R_t \subseteq \cup_{\forall a_i \in A_t} R_{a_i}$;

2) The predefined objective can be achieved by the task execution of A_t, such as minimizing the execution time [Ma et al. 1982] or maximizing the reliability [Shatz et al. 1992].

3) The agents in A_t can execute the allocated tasks under the constraint of the network structure, e.g. $\forall a_i, a_j \in A_t \Rightarrow P_{ij} \subseteq E$, where P_{ij} denotes the negotiation path between a_i and a_j.

If an agent has a higher probability of accessing the necessary resources for tasks, it may be allocated with more tasks. However, if too many tasks are crowded on certain agents with high probabilities of accessing the tasks' required resources, the tasks will require significantly more time to wait for the necessary resources [Liu et al. 2005; Jiang and Huang 2012; Jiang and Jiang 2009]. Moreover, the problem of waiting time may outweigh the advantage of the time saved by accessing resources at the allocated agents; therefore, we should now apply load balancing to the task allocation.

Definition 3.2 (*Load Balancing in Task Allocation*). Given a NMAS, $N =< A, E >$, where A is the set of agents, $\forall a_i \in A$, the team of tasks that queue for resource r_k of agent a_i can be denoted as Q_{ik}. The size of Q_{ik} is s_{ik} and the processing capability of a_i is v_i; the original probability of agent a_i to receive tasks (which need k type resources) is $P_i(k)$. We should perform load balancing when s_{ik} is too large, which is implemented by adjusting the probability of a_i's receiving new tasks:

$$P_i^{'}(k) = \psi(s_{ik}/v_i) \cdot P_i(k) \tag{1}$$

where ψ is an attenuation function, $0 \leq \psi \leq 1$; the value of $\psi(s_{ik}/v_i)$ *decreases monotonically from 1 to 0 as* s_{ik}/v_i *increases*.

3.1.2. Objectives of Task Allocation with Load Balancing in Unreliable NMASs. One of the main goals of task allocation is to minimize the task execution time [Liu et al. 2005; Shehory and Kraus 1998; An et al. 2011; Jiang and Jiang 2009; Ma et al. 1982; Chow and Kwok 2002; Xu et al. 2011]. Task execution in NMASs can be described as the operations of agents when accessing the necessary resources distributed in the networks [An et al. 2011; Weerdt et al. 2012; Jiang and Jiang 2009]. Therefore, *one of the key problems in reducing the execution time of a task is to reduce the time used to access the resources necessary for the task* [Jiang and Huang 2012; Jiang et al. 2013b; Chow and Kwok 2002; Xu et al. 2011; Hong et al. 2007]. In a NMAS, *the resource access time includes two factors* [Xu et al. 2011]: *1) the* ***communication time*** *between the allocated agents in the network*,$\sum_{\forall a_i, a_j \in A_t} C_{ij}$; and *2) the task's* ***waiting time for resources*** *at the agents*, $\sum_{\forall a_i \in A_t} W_{ti}$. ($A_t$ is the set of agents allocated to task t, C_{ij} is the communication time between a_t and a_j, and W_{ti} is the waiting time of task t at agent a_i.).

Moreover, if an allocated agent is unreliable in the NMAS and cannot provide the desired resources, the other allocated agents will require more time to seek the missing resources. A new task allocation may be implemented if the task cannot obtain the necessary resources, which will waste additional time. Therefore, we should **guarantee reliable resource access as well as minimize resource access time** to reduce task execution time in unreliable NMASs.

Thus, **the objective of task allocation with load balancing in unreliable N-MAS** is to select the agent set, A_t, to satisfy the following situation:

$$A_t = argmin_{A_t \subseteq A}((\overbrace{\underbrace{\sum_{\forall a_i, a_j \in A_t} C_{ij}}_{Task\ allocation}}^{Communication\ time} + \overbrace{\underbrace{\sum_{\forall a_i \in A_t} W_{ti}}_{Load\ balancing}}^{Waiting\ time})/ \overbrace{\underbrace{Rel(A_t)}_{Reliable\ allocation}}^{Reliability}) \quad (2)$$

where $Rel(A_t)$ is the probability that A_t can contribute reliable resources for task t. To reduce the communication time and waiting time, we should implement effective task allocation and load balancing; to improve the reliability, we should implement reliable task allocation.

3.2. New Problems Taken by Multiplex Networks

Multiplex networks arise when agents are connected by multiple types of links [Gómez-Gardeñes et al. 2012; Yağan and Gligor 2012; Szell et al. 2010; Brummitt et al. 2012], where each type of links may have a different relative bias and reliability for communicating a different type of resources. Each type of links and the involved agents comprise a network layer. In summary, the new problems of task allocation with load balancing in unreliable multiplex networks can be described as follows:

— The communication time between agents in the multiplex network is influenced by the link types along their communication paths. Therefore, C_{ij} in Equation (2) and the resource negotiation model between agents should be adjusted for multiple link types.
— Each network layer has different reliability for communicating different types of resources, and an agent may have different reliability when it is attributed to different network layers. Therefore, $Rel(A_t)$ in Equation (2) should consider the effects of network layers.
— The tasks may wait not only at individual agents but also at network layers that are composed of the same types of links and involved agents. Therefore, the waiting time in Equation (2) should also consider the factor of network layers.

To solve the above problems, we adopt the following measures in this paper:

— We devise a new model to negotiate resources between agents in multiplex networks. With this resource negotiation model, an agent's accessibility to a resource is determined by the link types as well as the communication distance between the agent and the resource.
— We present new definitions for the reliability of agents and network layers and devise a new reliable task allocation model based on both agents and network layers.
— We present a new load balancing mechanism, which adjusts the probabilities of agents and associated network layers to obtain new tasks if too many tasks are crowded on certain agents. The new load balancing mechanism can alleviate both the waiting time at heavily burdened agents and the waiting time at highly congested network layers.

4. MODELING MULTIPLEX NETWORKS

4.1. Network Layers in Multiplex Networks

Definition 4.1 (*Network layer in multiplex network*). In a multiplex network, the same type of links and involved agents comprise a **network layer**. Assume that the links in the multiplex social network, $N =< A, E >$, are classified into λ different types 1,..., λ. N can then be split to λ network layers. Each network layer, N_x, where

$1 \leq x \leq \lambda$, is defined as follows:

$$N = \{N_x |\ N_x =< A_x, E_x > \wedge A_x \subseteq A \wedge E_x \subseteq E \wedge (\forall e_{xi}, e_{xj} \in E_x \Rightarrow l_{e_{xi}} = l_{e_{xj}})\} \quad (3)$$

where e_{xi} and e_{xj} are the links in the network layer N_x, $l_{e_{xi}}$ and $l_{e_{xj}}$ denote the types of e_{xi} and e_{xj}.

In a multiplex network, each agent may associate with different network layers and other agents.

Definition 4.2 (*Associated network layers and agents*). Given a multiplex network, $N =< A, E >$; $\forall a_i \in A$, the associated network layers of a_i are defined as the set of network layers that have direct or indirect links with a_i. Thus, the associated network layers of a_i with d hops are defined as follows:

$$AN_{a_i}(d) = \{N_x | N_x \in N \wedge h(a_i, N_x) = d\} \quad (4)$$

where $h(a_i, N_x)$ denotes the hops from a_i to N_x:

$$h(a_i, N_x) = \min_{\forall a_j \in A_x} h(a_i, a_j) \quad (5)$$

where $h(a_i, a_j)$ denotes the hops between a_i and a_j in the network, and the hops between two adjacent agents is set to 1. Then, the d^{th}-order set of associated agents of a_i is:

$$AA_{a_i}(d) = \{A_x | \forall N_x \in AN_{a_i}(d)\} \quad (6)$$

In NMASs, some resources are placed at agents within the network and can be accessed by agents to execute tasks. A resource at an agent can be communicated and accessed by other agents. Based on the benchmark work in a previous study [Yağan and Gligor 2012], **the resource communication relevant to link types in multiplex networks** is set as follows in this paper:

Let there be m types of resources in the multiplex social network $N =< A, E >$; N can be split to λ network layers. Each network layer, $\forall N_x \in N$, is associated with a parameter, $c_{xk}(1 \leq x \leq \lambda, 1 \leq k \leq m)$, for k-type resources that measures the relative bias speed N_x has in communicating k-type resources. The value of c_{xk} negatively correlates with the communication time cost necessary for the k-type resources in N_x.

4.2. Tasks and Resource Negotiation in Multiplex Networks

Without the loss of generality, task execution in NMASs can be described via the operations of agents when accessing necessary resources distributed in the networks [An et al. 2011; Weerdt et al. 2012; Jiang and Jiang 2009; Jiang et al. 2013b]. Therefore, a task can be decomposed into $< operations, resources >$. Each task can be implemented if its required resources are all satisfied, or else it will wait for the resources. The execution of a task can be simply described as follows: 1) access the required resources (if the resources are occupied by other tasks, this task should wait until the resources are freed); 2) carry out the operations while all resources are satisfied; 3) free the occupied resources after the operations are finished. Let the set of operations of a task, t, be $OP_t = \{op_{t1}, op_{t2}, \ldots, op_{tn}\}$. The set of resources for each operation is R_{opi}, where $1 \leq i \leq n$. Thus, the total task execution time is defined as follows:

$$E_t = \sum_{op_{ti}} (exec\_time(op_{ti}) + resource\_access\_time(R_{op_{ti}})) \quad (7)$$

Therefore, the execution time of a task is composed of two parts, one is the real executing time of operations, and the other is the time for accessing resources.

The operations in tasks are related to real applications, and the operations in different real tasks may significantly vary. Therefore, this paper only focuses on the optimization of resource access in multiplex social networks (Eq.(2)), which is crucial to the performance of tasks in NMASs, as stated in Section 3.1.2.

When some agents attempt to access the resources of other agents to execute tasks, they should negotiate the resources in the multiplex networks; and resources of an agent can be communicated via the negotiation path to another agent.

We now design the algorithm to compute the resource negotiation path in multiplex networks. Let the multiplex network be $N =< A, E >= \{< A_x, E_x > |1 \leq x \leq \lambda\}$. The algorithm to compute the resource negotiation path in multiplex networks is described in Algorithm 1. We assume that the communication time of two adjacent agents within the same network layer, N_x, for k-type resources is $1/c_{xk}$. P_{ij}^k denotes the k-type resource negotiation path between a_i and a_j; C_{ij}^k denotes the communication time for accessing a k-type resource along P_{ij}^k, and $< a_i, a_j >_x$ denotes the x-type link between a_i and a_j.

Algorithm 1. Resource negotiation path between two agents for k-type resource in the multiplex networks.

/*Reduce the multiplex network to a weighted single layer network*/
For $(i = 1; i \leq |A|; i++)$
  For $(j = 1; j \leq |A|; j++)$
    $\{b = \infty; x^* = 0;$
    For $(x = 1; x \leq \lambda; x++)$
      {**If** there is a x-type link between a_i and a_j, **then**:
        {**If** $c_{xk} < b$, **then**:$\{b = c_{xk}; x^* = x\};\}\}$
    If $b \neq \infty$, **then**: $\{P_{ij}^k = \{< a_i, a_j >_{x^*}\}; C_{ij}^k = 1/b\}$
    else:$\{P_{ij}^k = \emptyset; C_{ij}^k = \infty\};\}$
/* Compute the shortest resource negotiation paths and minimum communication time costs */
For $(m = 1; m \leq |A|; m++)$
  For $(i = 1; i \leq |A|; i++)$
    For $(j = 1; j \leq |A|; j++)$
      If $C_{ij}^k > (C_{im}^k + C_{mj}^k)$,
      then:$\{P_{ij}^k = P_{im}^k \cup P_{mj}^k; C_{ij}^k = C_{im}^k + C_{mj}^k\};$
Output P and C.

4.3. Unreliable Situations in Multiplex Networks

In multiplex networks, each type of links may have different relative biases for communicating different types of resources [Yağan and Gligor 2012]. Thus, the communication of some resources on some network links may be unreliable. Moreover, some agents may take unreliable actions and cannot contribute reliable resources for executing tasks due to the openness and heterogeneity of NMASs [Weerdt et al. 2012; Ohtsuki et al. 2006]. Therefore, the unreliable situations to access resources in task execution in multiplex networks include two aspects: unreliable network links and unreliable agents.

4.3.1. Unreliable Network Links. The information loss rate is a very important performance measure for most network communication systems [Zhou et al. 2007]. Therefore, we also present the concept of the resource loss rate of x-type links for communicating k-type resources, L_x^k, to measure the reliability of links in multiplex networks:

$$L_x^k = (n_x^k - n_x^{k'})/n_x^k \tag{8}$$

where n_x^k is the total number of k-type resources communicated via the network layer, N_x (i.e. the x-type links); $n_x^{k'}$ is the number of k-type resources that are successfully communicated via N_x. The x-type links are *unreliable* for k-type resources if $L_x^k \neq 0$. The value of L_x^k negatively correlates with the reliability of x-type links for communicating k-type resources.

4.3.2. Unreliable Agents. In our previous work [Jiang et al. 2013b], we modeled the unreliable agents according to their behaviors toward resources during the periods of task allocation and task execution, which neglects the variation of resource types. We now describe the unreliable agents in a multiplex network by considering varying resource types.

Let there be m types of resources in a multiplex social network. The resource status of an agent in this network can then be described as $R_a = \sum_{1 \leq k \leq m} n_k r_k$, which denotes that the agent owns k-type resources $(1 \leq k \leq m)$ with the amount of n_k. We can now define the unreliable agents as follows:

Definition 4.3 (*Unreliable agent*). ***An agent is unreliable*** *if it satisfies one of the following conditions*:

(1) *It fabricates its resource status information during task allocation, which can be described as follows: Let there be an agent, a_i; the real resources owned by a_i are $R_{a_i} = \sum_{1 \leq k \leq m} n_k r_k$. When the task allocation heuristic inquiries into the resource status of a_i and the reported resources status of a_i is $MR_{a_i} = \sum_{1 \leq k \leq m} n_k^{'} r_k$, a_i is unreliable for its k-type resource status information if $n_k^{'} \neq n_k$.*
(2) *It does not contribute all its free resources during task execution if the allocated task requires it to do so, which can be described as follows: Let task t require some resources from a_i, which are denoted as $R_{a_i}^t = \sum_{1 \leq k \leq m} n_k^t r_k$. The set of resources that a_i really contributes to task t is $R_{a_i}^{t'} = \sum_{1 \leq k \leq m} n_k^{t'} r_k$. Let the real resources owned by a_i be $R_{a_i} = \sum_{1 \leq k \leq m} n_k r_k$. If $n_k^{t'} \neq n_{k^t} \wedge n_k^{t'} < n^k$, we can say that a_i is unreliable to contribute k-type resources during task execution.*

5. TASK ALLOCATION MODEL

5.1. Resource Accessibility of Agents and Network Layers

As stated above, our task allocation objective is to **guarantee reliable resource access and minimize resource access time by considering the characteristics of multiplex networks**, which includes two aspects: 1) tasks can receive reliable resources to be *successfully* executed in multiplex networks (i.e. the task's required resources can be satisfied reliably); and 2) tasks can receive necessary resources to be *efficiently* executed in multiplex networks (i.e. the resource access time can be reduced). For the first aspect, the tasks should be allocated to the reliable network layers and agents; for the second aspect, the tasks should be allocated to the network layers and agents such that resources can be accessed with less time costs.

5.1.1. Negotiation Reputation. In this paper, we use the negotiation reputation to measure the reliability of an agent or network layer to contribute real resources for executing allocated tasks.

Definition 5.1 (*Negotiation reputation of an agent*). In a multiplex network, each agent is associated with a weight, $w\_a_i(k)$, for k-type resources, which represents the negotiation reputation of a_i for accessing k-type resources. The **initial negotiation reputation** can be set according to the proportion of the number of k-type resources

owned by this agent to the average number of k-type resources of all agents in the network:

$$w\_a_i(k) = n_i(k) / \left(\frac{\sum_{\forall a_j \in A} n_j(k)}{|A|} \right) \tag{9}$$

where $n_i(k)$ (or $n_j(k)$) denotes the number of k-type resources of a_i (or a_j). In reality, $w\_a_i(k)$ should be adapted according to the execution results of tasks that require k-type resources from a_i ; e.g. if a task (needing k-type resources from a_i) is successfully completed, $w\_a_i(k)$ will be gained and vice versa. The detailed adaptation of $w\_a_i(k)$ is demonstrated by the reward mechanism in Section 5.3.

Definition 5.2 (*Negotiation reputation of a network layer*). The initial negotiation reputation of a network layer is determined by the initial negotiation reputations of all agents within the layer and its bias speed for communicating k-type resources, c_{xk}. Let the network layer be $N_x =< A_x, E_x >$; the **initial negotiation reputation** of N_x for k-type resources is defined as follows:

$$w\_N_x(k) = \alpha \cdot \sum_{\forall a_i \in A_x} w\_a_i(k) + (1-\alpha) \cdot \frac{c_{xk}}{(\sum_{\forall N_y \in N} c_{yk})/\lambda} \tag{10}$$

where $0 \leq \alpha \leq 1$; λ is the number of network link types, i.e. the number of network layers in a multiplex network, N. Because the numbers of agents in different network layers may significantly vary, the negotiation reputations of different network layers may differ significantly from each other. Thus, we now present the standardized $w\_N_x(k)$:

$$ws\_N_x(k) = w\_N_x(k) / \left(\frac{\sum_{\forall N_y \in N} w\_N_y(k)}{\lambda} \right) \tag{11}$$

The $ws\_N_x(k)$ should also be adapted according to the execution results of tasks that require k-type resources from N_x; e.g. if a task (needing k-type resources from N_x) is successfully completed, $ws\_N_x(k)$ will be gained and vice versa. The detailed adaptation of $ws\_N_x(k)$ is demonstrated by the reward mechanism in Section 5.3.

5.1.2. Resource Accessibility. To measure the probability of an agent or a network layer that *successfully and efficiently* receives resources for allocated tasks in a multiplex network, we now present the concept of resource accessibility, which includes two parts: 1) the reliability that true resources can be obtained, which can be measured by the negotiation reputations of agents and network layers; 2) the extent that the communication time required to access resources can be reduced, which can be measured by the time costs of resource negotiation paths. Moreover, because agents always negotiate resources with other associative agents, the resource accessibility of an agent or a network layer is also influenced by its associative agents or associative network layers.

Definition 5.3 (*Resource accessibility of an agent*). Let the multiplex social network be $N =< A, E >$; the accessibility of agent a_i for k-type resources is defined as follows:

$$\Re a_i(k) = \sum_{\forall a_j \in AA_{a_i}} \left(w\_a_j(k) \cdot \frac{n_j(k)}{(\sum_{\forall a_j \in A} n_j(k))/|A|} \cdot \frac{1}{C_{ij}^k} \right) + w\_a_i(k) \cdot \frac{n_i(k)}{(\sum_{\forall a_j \in A} n_j(k))/|A|} \cdot \frac{1}{C_*} \tag{12}$$

where C_{ij}^k denotes the time cost for negotiating k-type resources between agent a_i and a_j, which is calculated by Algorithm 1; $n_i(k)$ denotes the number of k-type resources of a_i; $|A|$ denotes the number of all agents in A; AA_{a_i} denotes the all associated agents of a_i; C_* is defined as $C_* < \min_{\forall a_j \in AA_{a_i}}(C_{ij}^k)$.

Definition 5.4 (*Resource accessibility of a network layer*). Let the multiplex social network be $N =< A, E >$; $N = \{N_x | \ 0 \leq x \leq 1\}$, where λ is the number of network layers; $N_x =< A_x, E_x >$. To reduce the communication time required by agents to negotiate resources in task execution within a network layer, we will select the agents in this network layer that are more compact; thus, the compactness of agents in the network layer positively correlates with the resource accessibility of the network layer. Therefore, the resource accessibility of N_x is determined by the negotiation reputation of N_x, the resource accessibilities of all agents within N_x, and the distribution of agent localities in N_x:

$$\Re N_x(k) = ws\_N_x(k) \cdot \sum_{\forall a_i \in A_x} \left(\Re a_i(k) \cdot \frac{1}{C^k_{ix^*}} \right) \tag{13}$$

where $C^k_{ix^*}$ denotes the time cost for negotiating k-type resources between agent a_i and a_{x^*}, and a_{x^*} is the center of network layer N_x for k-type resources:

$$a_{x^*} = argmin_{\forall a_i \in A_x} (\sum_{\forall a_j \in (A_x - \{a_i\})} C^k_{ij}) \tag{14}$$

5.2. Task Allocation

5.2.1. ***Introduction of Allocation Architecture.*** In our previous work [Jiang and Jiang 2009; Jiang et al. 2013b], we presented a spectrum that ranged from a totally centralized approach to a totally decentralized approach to task allocation based on the traditional manger/contractor allocation architecture of the Contract Net Protocol [Aknine et al. 2004]: the centralized heuristic is utilized to control the overall status information, and the distributed heuristic is utilized to achieve the flexibility of task allocation. The presented task allocation process in our previous work [Jiang and Jiang 2009; Jiang et al. 2013b] can be described as follows. A task may be first allocated to one agent using a centralized heuristic. The agent then takes charge of the execution of the task (we call this agent the *manager*). When the manager lacks the necessary resources to execute the allocated task, it negotiates with other agents in the network for resource assistance using a distributed heuristic; if other agents have the required resources (we call the agents that provide resources to the manager *contractors*), the manager and contractors will work together to execute the task.

In the above approach, the manager is fixed throughout the allocation process of a task, which is called the *fixed manager method*; all contractors are selected by the manager, thus it mainly optimizes the communication between the manger and contractors and does not optimize the communication between all allocated agents.

We now substantially extend the architectures proposed in previous studies [Jiang and Jiang 2009; Jiang et al. 2013b; Aknine et al. 2004] by presenting a new method in which the manger is not fixed during the task allocation process. This approach is called the *alterable manger method* and can be briefly explained as follows. At first, a manager is selected for a task by using a centralized heuristic; this manager will then seek another agent to act as a contractor to obtain the highest resource accessibility that can satisfy the resource requirements of a task. Next, these two agents become the already allocated agents. *Each agent in the group of already allocated agents will then act as a manager to seek the next contractor* to obtain the highest resource accessibility that will satisfy the resource requirements of the task. Finally, the optimal contractor can be allocated, and the new already allocated agents will seek the following contractor again. This process will repeat until all resources required by the task can be satisfied or all agents are allocated. Therefore, the main novelty of the alterable manager method is that each agent among the already allocated agents for a task can act as a manager. Thus, this model outperforms the traditional fixed manager method by allowing agents to use the results of the previously allocated agents when seeking

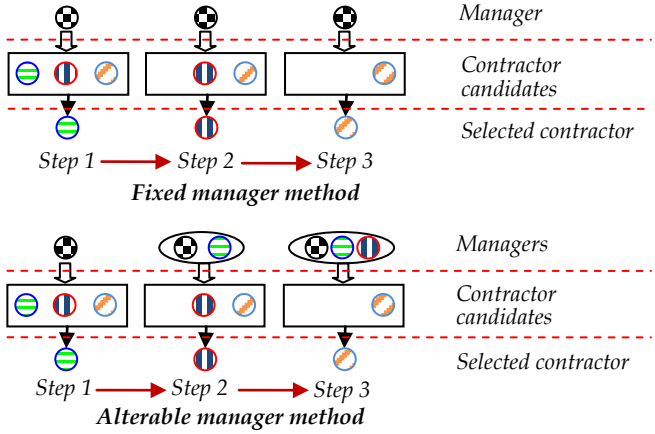


Fig. 1. Fixed manager method and alterable manager method in task allocation

an optimal result. Fig. 1 illustrates the difference between the fixed manager method and the alterable manager method.

Moreover, to consider the characteristics of multiplex networks, the targets of task allocation include network layers as well as agents, i.e. *the managers and contractors may be network layers or agents*. Therefore, we now present two allocation models, *network layer-oriented allocation and agent-oriented allocation.* In the network layer-oriented allocation, the managers and contractors are all network layers; the network layers satisfying the resource requirements of a task are first allocated, and the final agents will then be selected from the allocated network layers. However, in the agent-oriented allocation, the mangers and contractors are agents that are directly selected from all of the agents in the whole network. *These two mechanisms both consider the link characteristics of multiplex networks*.

5.2.2. ***Network Layer-Oriented Allocation for Multiplex Networks***.

1) Allocation of Network Layers

Definition 5.5 (*Distance between two network layers*). Given a multiplex network, $N =< A, E >$; $\exists N_x$, $N_y \in N$, the negotiation distance between N_x and N_y for k-type resources is defined as follows:

$$D^k_{xy} = C^k_{x^*y^*} \tag{15}$$

where $C^k_{x^*y^*}$ denotes the communication time cost for k-type resource negotiation between agents a_{x^*} and a_{y^*}, which is calculated according to Algorithm 1; a_{x^*} and a_{y^*} are the centers of network layers N_x and N_y, respectively, which can be determined according to Equation (14).

Let R_t be the set of resources required by task t, $\overline{R_t}$ the set of resources for task t that are currently lacking, and R_{N_x} the set of resources that owned by network layer N_x. The resources that N_x may contribute to task t are then defined as $\overline{R_t} \cap R_{N_x}$; $|\overline{R_t} \cap R_{N_x}|_k$ denotes the number of k-type resources in $\overline{R_t} \cap R_{N_x}$. Let the manager network layer be N_x, and let task t require m_t types of resources. If N_x attempts to select the contractor

from other network layers (e.g. N_y), N_x will observe the *negotiation value* of N_y:

$$VN_y(t) = \sum_{1 \le k \le m_t} (\Re N_y(k) \cdot |\overline{R_t} \cap R_{N_x}|_k / D^k_{xy}) \tag{16}$$

The manager selects the contractor from the candidates according to their negotiation values arranged in descending order.

Theorem 1. *It is assumed that task is t and the negotiation values are correct. Let the manager be N_x, and let the two contractor candidates be N_y and N_z; $S(N)$ denotes the degree that the task allocation objective can be satisfied by N. Therefore: $VN_y(t) > VN_z(t) \Rightarrow S(\{N_x\} \cup \{N_y\}) > S(\{N_x\} \cup \{N_z\})$.*

Proof sketch. *According to Equation (2), if we do not consider the task's waiting time for resources which is influenced by load balancing, $S(N)$ is determined by two factors: 1) the negotiation distance between the network layers in N; and 2) the reliability that N can contribute real resources. In Equation (16), $\Re N_y(k)$ includes the negotiation reputation of the network layer that can measure the reliability of the contractor candidate; $|\overline{R_t} \cap R_{N_x}|_k / D^k_{xy}$ is inversely proportional to the negotiation distance between the allocated network layers. Therefore, $VN_y(t) > VN_z(t)$ denotes that the value of the negotiation reputation divided by total communication time costs of $N_x \cup N_y$ is higher than that of $N_x \cup N_z$. Thus, $S(N_x \cup N_y) > S(N_x \cup N_z)$.*

Therefore, the negotiation value in Equation (16) can be used to satisfy the task allocation objectives in Equation (2) based on Theorem 1. We now use the alterable manager method to implement network layer-oriented task allocation, as shown in Algorithm 2.

Algorithm 2. Network layer-oriented task allocation.

1) $N_* = \arg\max_{\forall N_x \in N}(\Re N_x(k))$; /*$k$-type resources are the ones that task t needs mostly*/
2) $\overline{R_t} = R_t - R_{N_*}$;$N' = N - \{N_*\}$;$N_t = \{N_*\}$; $b1 = 0$; $b2 = 0$; $n = 0$;
3) **If** $\overline{R_t} == \emptyset$, **then**: $\{b1 = 1\}$;
4) **While** (($b1 == 0$ and $b2 == 0$)) **do**:
  4.1) $max = 0$; $b2 = 1$;
  4.2) $\forall N_y \in N'$:
    4.2.1) $max_{temp} = 0$;
    4.2.2) $\forall N_x \in N_t$;
      4.2.2.1) $VN_y(t) = \sum_{1 \le k \le m_t}(\Re N_y(k) \cdot |\overline{R_t} \cap R_{N_x}|_k / D^k_{xy})$;
      4.2.2.2) **If** $VN_y(t) > max_{temp}$, **then**:$\{max_{temp} = VN_y(t)\}$; /*Now N_x is the manager*/
    4.2.3) **If** $max_{temp} > max$, **then**:$\{max = \max_{temp}; b2 = 0; N_{temp} = N_y;\}$
  4.3) **If** $(b2 == 0)$, **then**:$\{N_t = N_t \cup \{N_{temp}\}$; $\overline{R_t} = \overline{R_t} - R_{Ntemp}$;
    $N' = N' - \{N_{temp}\}$; $n++$; $N_{tn} = N_{temp}$;$\}$
  4.4) **If** $\overline{R_t} == \emptyset$, **then**: $\{b1 = 1\}$;
5) **If** $(b1 == 1)$, **then Return** (N_t);
    else Return $(False)$;
6) **End**.

Algorithm 2 is $O(|R_t| \cdot \lambda^2)$, where λ is the number of network layers.

Let there be a set of network layers N. The resource accessibility of N for k-type resources can be defined as $\Re N(k) = \max_{\forall N_x \in N}(\Re N_x(k))$. If the task is t; the set of allocated network layers by using a network layer-oriented allocation mechanism with alterable manager manner is N_t, $N_t \subseteq N$; and the set of allocated network layers by using the network layer-oriented allocation mechanism with fixed manager method is N'_t, $N'_t \subseteq N$.Thus, the following theorem can be derived:

Theorem 2. *Let the multiplex social network be $N = \langle A, E\rangle$, $N = \{N_x | \ 1 \leq x \leq \ \lambda\}$, where λ is the number of network layers. If a task, t, needs k-type resources, $\Re N_t(k) \geq \Re N'_t(k)$.*

Proof sketch. *With the fixed manager method, the manager will seek the contractor with the highest resource accessibility for k-type resources from the viewpoint of this fixed manager in each allocation step. Thus, only the resource accessibility of the contractor and the communication time cost between the manager and contractor can be optimized. With the alterable manager method, each one in already allocated network layers will act as a manager to seek the contractor with the highest resource accessibility for k-type resources from the manager's viewpoint. Finally, the contractor candidate with the highest negotiation value from all managers will be selected. Thus, the resource accessibility of a contractor and the minimum communication time cost between the contractor and the already allocated network layers are optimized. Therefore, $\Re N_t(k) \geq \Re N'_t(k)$.* □

Theorem 2 proves that our presented alterable manager allocation architecture outperforms the previous fixed manager allocation architecture by improving the resource accessibility of allocated network layers.

Theorem 3. *Let the set of allocated network layers using Algorithm 2 be N_t and the first manager be N_*. Another set of network layers, N',is then assumed to include N_* and satisfy all the resources in R_t:*

$$(\forall N' \wedge (N_* \in N') \wedge (N' \subseteq N) \wedge (R_t \subseteq R_{N'})) \Rightarrow \sum_{\forall N_y \in (N_t - \{N_*\})} VN_y(t) \geq \sum_{\forall N_y \in (N' - \{N_*\})} VN_y(t)$$

Proof sketch. *We can now use reductio ad absurdum to prove Theorem 3. Assume a set of network layers N', $N_* \in N' \wedge N' \neq N_t$, that can provide all the required resources for executing task t, and the total negotiation values of $N' - \{N_*\}$ by using the alterable manager manner is $\sum_{\forall N_y \in (N' - \{N_*\})} VN_y(t)$; if $\sum_{\forall N_y \in (N_t - \{N_*\})} VN_y(t) < \sum_{\forall N_y \in (N' - \{N_*\})} VN_y(t)$, there are network layers with lower negotiation values that can provide the required resources in $\overline{R_t}$ and be selected by the already allocated network layers in Algorithm 2. However, the higher negotiation-value network layers with the required resources in $\overline{R_t}$ are not selected by the already allocated network layers. In Algorithm 2, the selection is implemented by Step 4.2 and 4.2.2, which guarantees that the selected network layer in each Step 4.2 has the maximum negotiation value from the currently already allocated network layers. Therefore, a situation in which $\sum_{\forall N_y \in (N_t - \{N_*\})} VN_y(t) < \sum_{\forall N_y \in (N' - \{N_*\})} VN_y(t)$ cannot occur in Algorithm 3. Theorem 3 addresses this issue.* □

Based on Theorem 3, Algorithm 2 can find the network layers with the maximum negotiation values, thus satisfying the objectives of task allocation in Equation (2) according to Theorem 1.

2) Select Final Agents from Allocated Network Layers

After the network layers are allocated by using Algorithm 2 for a task, the real agents within the allocated network layers will be selected to execute the task. In each allocated network layer, an initiator agent with the highest resource accessibility is first selected, and this agent will then negotiate with other agents from near to far within the network layer until all required resources are satisfied or all agents within the

network layer are considered. The selection of agents within allocated network layers is demonstrated by Algorithm 3.

Algorithm 3. Selecting final agents from allocated network layers.
/$N_t = \{N_{tx} | N_{tx} = \langle A_{tx}, E_{tx} \rangle, 1 \leq x \leq n\}$, are the allocated network layers resulted from Algorithm 2 */*

1) $x = 1; b = 0; \overline{R_t} = R_t$;
2) **While**(($x \leq n$) and $b == 0$) **do**:
  2.1) **Set** the tags for all agents in A_{tx} to 0 initially;
  2.2) $a_{x^*} = \arg\max_{\forall a_i \in A_{tx}}(\Re a_i(k))$;
  2.3) **Create** Queue (Q_x); **Insert** Queue (Q_x, a_{x^*}); **Set** the tag of a_{x^*} to 1;
  2.4) $A_t = \{a_{x^*}\}$;$\overline{R_t} = \overline{R_t} - R_{ax^*}$;
  2.5) **If** $\overline{R_t} == \emptyset$, **then**: $\{b = 1\}$;
  2.6) **While** ((!EmptyQueue(Q_x))and ($b == 0$)) **do**:
    2.6.1) $aout$ =**Out** Queue(Q_x);$R' = \overline{R_t} - R_{aout}$;
    2.6.2) **If** $R' \neq \overline{R_t}$, **then**:$\{\overline{R_t} = \overline{R_t} - R_{aout}$; $A_t = A_t \cup \{aout\}\}$;
    2.6.3) **If** $\overline{R_t} == \emptyset$, **then**:$\{b = 1\}$;
    2.6.4) $\forall alocal \in L_{aout}$: */*$L_{aout}$ is the set of neighbors of $aout$ in N_{tx}*/*
      If the tag of $alocal$ is 0, **then**:{**Insert** Queue($Q_x, alocal$); **Set** the tag of $alocal$ to 1};
  2.7) $x++$;
3) **Return** (A_t);
4) **End**.

Algorithm 3 is $O(|N_t| \cdot |A_{tx}|)$, where $|N_t|$ is the number of network layers in N_t, $N_t = \{N_{tx} | N_{tx} =< A_{tx}, E_{tx} >\}$. **Therefore, the total time complexity of algorithms for network layer-oriented task allocation (Algorithm 2+Algorithm 3) is** $O(|R_t| \cdot \lambda^2)$**.**

Lemma 1. *Let the set of allocated agents in network layer N_{tx} using Algorithm 3 be A^*_{tx}, and let the initiator agent be a_{x^*}; the set of lacking resources of a_{x^*} to implement t is $\overline{R^t_{a_{x^*}}}$. It is, then, assumed that there is another set of agents in N_{tx}, A'_{tx}, that includes a_{x^*} and can also satisfy the resource requirements of t in N_{tx}; $Com_x(a_i, a_j)$ denotes the communication time between a_i and a_j within N_{tx}. Thus,*

$$(\forall A'_{tx} \wedge (A'_{tx} \subseteq A_{tx}) \wedge (\overline{R^t_{a_{x^*}}} \cap R_{A'_{tx}} = \overline{R^t_{a_{x^*}}} \cap R_{A^*_{tx}})) \Rightarrow \sum_{\forall a_i \in (A'_{tx} - \{a_{x^*}\})} Com_x(a_{x^*}, a_i) \geq \sum_{\forall a_i \in (A^*_{tx} - \{a_{x^*}\})} Com_x(a_{x^*}, a_i)$$

Proof. *If Algorithm 3 is used, the set of allocated agents in an allocated network layer N_{tx} is A^*_{tx}, and the total communication costs between a_{x^*} and other agents in $A^*_{tx} - \{a_{x^*}\}$ within N_{tx} are $\sum_{\forall a_i \in (A^*_{tx} - \{a_{x^*}\})} Com_x(a_{x^*}, a_i)$. Now, if there is a set of agents A'_{tx}, $A'_{tx} \neq A^*_{tx}$, which can provide the same set of resources to t as A^*_{tx}, and the total communication cost between a_{x^*} and other agents in $A'_{tx} - \{a_{x^*}\}$ is $\sum_{\forall a_i \in (A'_{tx} - \{a_{x^*}\})} Com_x(a_{x^*}, a_i)$; if $\sum_{\forall a_i \in (A'_{tx} - \{a_{x^*}\})} Com_x(a_{x^*}, a_i) < \sum_{\forall a_i \in (A^*_{tx} - \{a_{x^*}\})} Com_x(a_{x^*}, a_i)$, it denotes that there are any agents with farther distance that provide the required resources in $\overline{R^t_{a_{x^*}}}$, but the nearer agents with required resources do not provide the resources in $\overline{R^t_{a_{x^*}}}$. Obviously, such situation cannot take place in Algorithm 3 where a_{x^*} negotiates with other agents from near to far within the network layer as Step 2.6. Therefore, we have Lemma 1.* □

Therefore, Algorithm 3 can provide the initiator agent with the highest resource accessibility and the minimum communication time cost between this initiator agent and other allocated agents within the same network layer based on Lemma 1.

5.2.3. ***Agent-Oriented Allocation for Multiplex Networks.*** In agent-oriented allocation, the agents are allocated directly from the whole network. We also use the alterable manager method to allocate agents. Let a_i be the manager agent, and let task t require m_t types of resources. The *negotiation value* of a_j by a_i for task t can then be defined as follows:

$$Va_j(t) = \sum_{1 \le k \le m_t} (\Re a_j(k) \cdot |\overline{R_t} \cap R_{a_j}|_k / C_{ij}^k) \tag{17}$$

where $|\overline{R_t} \cap R_{a_j}|_k$ denotes the number of k-type resources in $\overline{R_t} \cap R_{a_j}$.

Theorem 4.*The task is assumed to be t, and the negotiation values are assumed correct. Let the manager be a_i, and let the two contractor candidates be a_j and a_k; $S(A)$ denotes the degree that the task allocation objective can be satisfied by agent set A. Thus, $V_{a_j}(t) > V_{a_k}(t) \Rightarrow S(\{a_i\} \cup \{a_j\}) > S(\{a_i\} \cup \{a_k\})$.*

Proof. *The proof is similar to that of Theorem 1.Thus, we omit the proof in the interest of brevity.* □

Therefore, the negotiation value in Equation (17) can be used to satisfy the task allocation objectives in Equation (2) from Theorem 4. In agent-oriented task allocation, the agent with the highest resource accessibility in the whole network is first allocated. The alterable manager method is then used to seek other agents to allocate; the task allocation process can be demonstrated by Algorithm 4.

Algorithm 4. Agent-oriented task allocation.
/*Let k-type resources be the ones that the task needs mostly */

---

1) $\overline{R_t} = R_t$;
2) a_*=arg $\max_{\forall a_i \in A}(\Re_{a_i}(k))$;
3) $\overline{R_t} = \overline{R_t} - R_{a_*}$; $A^{'} = A - \{a_*\}$; $A_t = \{a_*\}$;$b1 = 0$; $b2 = 0$;
4) **If** $\overline{R_t} == \emptyset$, **then**:$\{b1 = 1\}$;
5) **While** (($b1 == 0$) and ($b2 == 0$)) **do**:
  5.1) $max = 0$; $b2 = 1$;
  5.2) $\forall a_j \in A^{'}$:
    5.2.1) $max_{temp} = 0$;
    5.2.2) $\forall a_i \in A_t$:
      5.2.2.1) $Va_j(t) = \sum_{1 \le k \le m_t}(\Re a_j(k) \cdot |\overline{R_t} \cap R_{a_j}|_k / C_{ij}^k)$;
      5.2.2.2) **If** $Va_j(t) > max_{temp}$, **then**:$\{max_{temp} = Va_j(t)\}$;
        /*Now a_i is the manager agent*/
    5.2.3) **If** $max_{temp} > max$, **then**: $\{max = max_{temp}; b2 = 0; a_{temp} = a_j;\}$
  5.3) **If** ($b2 == 0$), **then**: $\{A_t = A_t \cup \{a_{temp}\}; \overline{R_t} = \overline{R_t} - R_{a_{temp}}; A^{'} = A - \{a_{temp}\};\}$
  5.4) **If** $\overline{R_t} == \emptyset$, **then**:$\{b1 = 1\}$;
6) **If** ($b1 == 0$), **then Return**(A_t);
    else Return(False);
7) **End**.

Algorithm 4 is $O(|R_t| \cdot |A|^2)$. **In real multiplex networks, $|A| >> \lambda$; thus, the real time cost of agent-oriented task allocation, $O(|R_t| \cdot |A|^2)$, may often be higher than that of network layer-oriented task allocation, $O(|R_t| \cdot \lambda^2)$.**

Theorem 5. *Let there be a set of agents A. The resource accessibility of A to k-type resources can be defined as: $\Re A(k) = max_{\forall a_i \in A}(\Re_{a_i}(k))$. If the task is t, the set of allocated agents by using the agent-oriented allocation mechanism with the alterable manager method is A_t, $A_t \subseteq A$, and the set of allocated agents by using the agent-oriented allocation mechanism with the fixed manager method is $A_t^{'}$, $A_t^{'} \subseteq A$. Thus,$\Re A_t(k) \ge \Re A_t^{'}(k)$.*

Proof. *The proof is similar to that of Theorem 2. Thus, we omit the proof in the interest of brevity.* □

Theorem 6. *Let the set of allocated agents using Algorithm 4 be A_t and let the first manager be a_*. It is, then, assumed that there is another set of agents, $A_t^{'}$, that includes a_* and can also satisfy all the resources in R_t. Thus,*

$$(\forall A_t' \wedge (a_* \in A_t') \wedge (A_t' \subseteq A) \wedge (R_t \subseteq R_{A_t'})) \Rightarrow \sum_{\forall a_j \in (A_t - \{a_*\})} Va_j(t) \geq \sum_{\forall a_j \in (A_t' - \{a_*\})} Va_j(t)$$

Proof. *The proof is similar to that of Theorem 3. Thus, we omit the proof in the interest of brevity.* □

Based on Theorem 6, Algorithm 4 can find the set of agents with the maximum negotiation values, which satisfies the objectives of task allocation in Equation (2) according to Theorem 4.

5.3. Reward Mechanism

To encourage network layers and agents in multiplex networks to provide reliable resources, we now present the reward mechanism.

5.3.1. Reward in Network Layer-Oriented Allocation. Each task is associated with a value; if the task can be successfully executed, the allocated network layers and agents will be rewarded with this value, or else they will be punished with this value. Let the associated value for task t be τ_t $(0 \leq \tau_t \leq 1)$, let N_t be the network layers allocated to t, and A_t be the set of agents that are really allocated to t. τ_t will then be divided into three parts: 1) reward to the allocated network layers (α); 2) reward to the allocated agents within the allocated network layers (β); and 3) reward to the involved agents that are out of allocated network layers but provide communication relay services for the resource negotiations between allocated network layers (γ). We can define $\alpha + \beta + \gamma = 1$, where $\gamma \ll \alpha + \beta$. The allocated network layers and agents will be rewarded (or punished) according to their real resource contributions.

The reward mechanism can then be defined as follows when the task is executed successfully:

$$\forall r_k \in R_t : \forall N_x \in N_t \Rightarrow ws\_N_x(k) = ws\_N_x(k) \cdot (1 + \alpha \cdot \tau_t \cdot (|R^t_{N_x}(k)|/|R_t|)) \tag{18}$$

where $R^t_{N_x}(k)$ is the set of k-type resources that N_x really contributes to task t in the execution.

$$\forall r_k \in R_t : \forall a_i \in A_t \Rightarrow w\_a_i(k) = w\_a_i(k) \cdot (1 + \beta \cdot \tau_t \cdot (|R^t_{a_i}(k)|/|R_t|)) \tag{19}$$

where $R^t_{a_i}(k)$ is the set of k-type resources that a_i really contributes to task t in the execution.

$$\begin{aligned} \forall r_k \in R_t : (\forall a_m \notin \cup_{N_x \in N_t} A_x) \wedge (\exists a_i, a_j \in A_t \wedge a_m \ \ is \ \ in \ \ P^k_{ij}) \\ \Rightarrow \ \ w\_a_m(k) = w\_a_m(k) \cdot (1 + \gamma \cdot \tau_t) \end{aligned} \tag{20}$$

Furthermore, the penalty mechanism can be defined as follows if the task is unsuccessfully executed. A new allocation should now be implemented.

$$\forall r_k \in R_t : \forall N_x \in N_t \Rightarrow ws\_N_x(k) = ws\_N_x(k) \cdot (1 - \alpha \cdot \tau_t \cdot (1 - |R^t_{N_x}(k)|/|R_t|)) \tag{21}$$

$$\forall r_k \in R_t : \forall a_i \in A_t \Rightarrow w\_a_i(k) = w\_a_i(k) \cdot (1 - \beta \cdot \tau_t \cdot (1 - |R^t_{a_i}(k)|/|R_t|)) \tag{22}$$

$$\begin{aligned} \forall r_k \in R_t : (\forall a_m \notin \cup_{N_x \in N_t} A_x) \wedge (\exists a_i, a_j \in A_t \wedge a_m \ \ is \ \ in \ \ P^k_{ij}) \\ \Rightarrow \ \ w\_a_m(k) = w\_a_m(k) \cdot (1 - \gamma \cdot \tau_t) \end{aligned} \tag{23}$$

5.3.2. Reward in Agent-Oriented Allocation. Let the associated value for task t be τ_t $(0 \leq \tau_t \leq 1)$, and A_t be the set of agents that are really allocated to t. τ_t will then be divided into two parts: 1) reward to the allocated agents (α); and 2) reward to the agents that provide communication relay services for the resource negotiations between allocated agents (β). We can define $\alpha + \beta = 1; \beta << \alpha$.

The reward mechanism can then be defined as follows when the task is executed successfully:

$$\forall r_k \in R_t : \forall a_i \in A_t \Rightarrow w\_a_i(k) = w\_a_i(k) \cdot (1 + \alpha \cdot \tau_t \cdot (|R^t_{a_i}(k)|/|R_t|)) \tag{24}$$

where $R^t_{a_i}(k)$ is the set of k-type resources that a_i really contributes to task t in the execution.

$$\begin{aligned} \forall r_k \in R_t : (\forall a_m \in A) \wedge (\exists a_i, a_j \in A_t \wedge a_m \ \ is \ \ in \ \ P^k_{ij}) \\ \Rightarrow \ \ w\_a_m(k) = w\_a_m(k) \cdot (1 + \beta \cdot \tau_t) \end{aligned} \tag{25}$$

The punishment mechanism can be defined as follows if the task is unsuccessfully executed. A new allocation should now be implemented.

$$\forall r_k \in R_t : \forall a_i \in A_t \Rightarrow w\_a_i(k) = w\_a_i(k) \cdot (1 - \alpha \cdot \tau_t \cdot (1 - |R^t_{a_i}(k)|/|R_t|)) \tag{26}$$

$$\begin{aligned} \forall r_k \in R_t : (\forall a_m \in A) \wedge (\exists a_i, a_j \in A_t \wedge a_m \ \ is \ \ in \ \ P^k_{ij}) \\ \Rightarrow \ \ w\_a_m(k) = w\_a_m(k) \cdot (1 - \beta \cdot \tau_t) \end{aligned} \tag{27}$$

5.4. Load Balancing in Task Allocation

As noted above, a network layer or agent may act as the manager or contractor for more tasks if it has a higher negotiation reputation or negotiation value. However, the tasks will wait significantly longer for the necessary resources if too many tasks are crowded on certain network layers or agents with high negotiation reputations or negotiation values [Liu et al. 2005; Jiang and Huang 2012; Jiang and Jiang 2009]. Moreover, the problem of waiting time may outweigh the advantage of the time saved by accessing resources at the allocated network layers and agents; therefore, we should now apply load balancing to the task allocation.

In previous related studies [Liu et al. 2005; Jiang and Huang 2012; Jiang and Jiang 2009; Jiang et al. 2013b], load balancing for an agent was implemented only based on the own load status of this agent, i.e. some tasks of an agent will be switched to other agents only when this agent's queuing tasks are large; this type of load balancing can be called *direct load balancing*. However, in multiplex networks, each agent or network layer is associated with other agents and network layers, and agents often negotiate with each other for resources to execute tasks. Therefore, the contextual load status (i.e. the load status of associated agents or network layers) needs to also be considered when we implement load balancing for an agent or network layer, which is called *contextual load balancing*.

Let the set of final allocated agents for task t be A_t. The team of tasks that queue for resource r_k of agent a_i can be denoted as Q_{ik}; the processing rate of a_i is v_i and the size of Q_{ik} is s_{ik}; $\psi_1(x)$ and $\psi_2(x)$ are two attenuation functions, $0 \leq \psi_1(x), \psi_2(x) \leq 1$, the values of $\psi_1(x)$ and $\psi_2(x)$ decrease monotonically from 1 to 0 as x increases.

For multiplex networks, we should perform load balancing respectively for the network layers and agents. First, we present the contextual load balancing for the network layers. Let the multiplex social network be $N =< A, E >$,$N = \{N_x | 1 \leq x \leq \lambda\}$. $L\_N^k_x$ and $CL\_N^k_x$ denote the own load status and contextual load status of the network layer N_x on resource r_k. We can balance the load for N_x by adjusting N_x's resource ac-

cessibility to r_k, $\Re N_x(k)$, as follows:

$$L\_N_x^k = \max_{\forall a_i \in A_t \wedge \forall a_i \in A_x} (s_{ik}/v_i) \tag{28}$$

$$CL\_N_x^k = \frac{\sum_{N_y \in \{N - N_x\}} (max_{\forall a_j \in A_t \wedge \forall a_j \in A_y} (s_{jk}/v_j)/D_{xy}^k)}{|N| - 1} \tag{29}$$

$$\Re N_x(k)' = \psi_1(L\_N_x^k + CL\_N_x^k) \cdot \Re N_x(k) \tag{30}$$

where D_{xy}^k is the distance between network layers N_x and N_y for negotiating r_k, which is calculated according to Equation (15).

Let there be an agent, a_i; $L\_a_i^k$ and $CL\_a_i^k$ denote the own load status and contextual load status of agent a_i on resource r_k. The contextual load balancing for agent a_i is then implemented by adjusting a_i's resource accessibility to r_k, $\Re a_i(k)$ as follows:

$$L\_a_i^k = s_{ik}/v_i \tag{31}$$

$$CL\_a_i^k = \frac{\sum_{a_j \in (A - \{a_i\})} (s_{jk}/(v_j \cdot C_{ij}^k))}{|A| - 1} \tag{32}$$

$$\Re a_i(k)' = \psi_2(L\_a_i^k + CL\_a_i^k) \cdot \Re a_i(k) \tag{33}$$

where C_{ij}^k denotes the negotiation distance between a_i and a_j for k-type resources, which is calculated according to Algorithm 1.

6. EXPERIMENTAL VALIDATION AND ANALYSES

6.1. Experimental Settings

The performance indices used in the experiments are presented as follows:

— *Success rate (sr)*. The success rate demonstrates the reliability of task allocation:

$$sr = (\sum_{i=1}^{n} (1/\xi_{t_i}))/n \tag{34}$$

where n is the number of total tasks; ξ_{t_i} indicates that the first $(\xi_{t_i} - 1)$ attempts to allocate task t_i are unsuccessful and only the ${\xi_{t_i}}^{\text{th}}$ allocation of t_i is successful.

— *Time costs:* 1) the total time costs (T), which are the sum of all time costs of all tasks and are measured from the initiation of the first task until the completion of the last task; 2) the communication time costs of all tasks (T_c), which are the sum of the communication time costs of all tasks for resource access that are influenced by both the distance between agents and the link types; 3) the waiting time costs of all tasks (T_w), which are the sum of the waiting time costs of all tasks that are determined by the waiting queue of the tasks on network layers and agents and their processing rates.

To validate the effectiveness of our models, we compare our presented multiplex network-adapted models, network layer-oriented model and agent-oriented model, with the following benchmark approaches:

— *Traditional resource-based task allocation model for simplex networks (Traditional simplex network-adapted model)*: the task will be allocated to the agents with a larger amount of the resources required by the task [Liu et al. 2005; Jiang and Huang 2012; An et al. 2011; Weerdt et al. 2012; Jiang and Jiang 2009]. This model

ignores the network link types and does not consider the biases of network links for communicating different types of resources.

— *Transparent task allocation model (Transparent model)*: the model can detect all deceptive agents in the network, and can also negotiate with the truthful agents through the reliable path with the lowest communication costs [Jiang et al. 2013b]. Although this model is not practical in real-world scenarios, it can be used as a benchmark to evaluate the task allocation performance in unreliable situations for multiplex networks.

Each experiment comprises 100 runs to obtain the average results. The simulation platform for experiments is developed in Java using the Eclipse IDE. The initial network is constructed by a random network model, in which 100 agents are included. The connection probability for any two randomly selected agents is set to 0.05. The initial network is then divided into 10 network layers with different communication speeds for different resources, and the ratio of the maximum speed to the minimum speed is 10. Each agent can be included in a network layer with a probability of 60 percent. There are eight types of resources to be available to the agents and tasks; each agent or task can only have four types of resources; the number of each type of resources of agents is set randomly, ranging from 1 to 9 with an average of 5, and the number of each type of resources of tasks is set randomly, ranging from 5 to 20 with an average of 12.5 [Jiang et al. 2013b]. For each run of the experiments, we set a total number of 500 tasks to be allocated by the system. (Note that there is one exception: in the robustness test, in order to show the properties of our models clearly, the total number of tasks is set to 2000.)

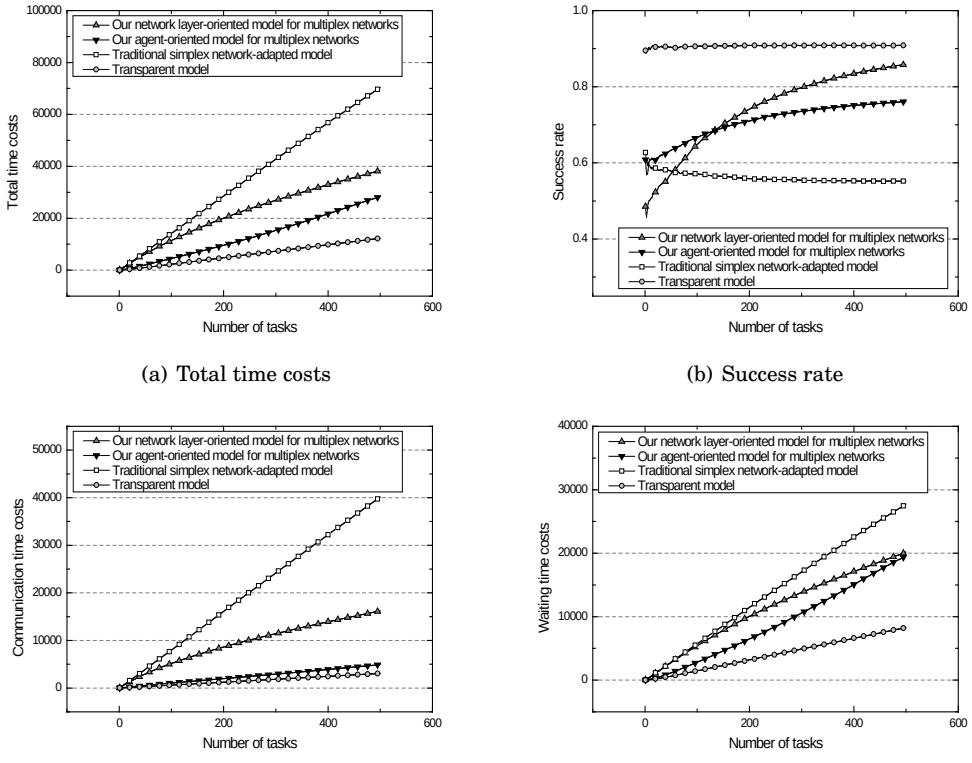
6.2. Results and Analyses

We analyze our models via the tests from the following three perspectives:

(1) The effectiveness of our models (the network layer-oriented model and the agent-oriented model for multiplex networks), which can be evaluated by comparing our model with several benchmark models introduced in Section 6.1.
(2) The effects of the following key components in our models: a) the reputation and reward mechanism; b) the consideration of multiple network link types in resource negotiation; and c) the contextual load balancing mechanism.
(3) The performance comparison between our presented two task allocation models for multiplex networks (the network layer-oriented model vs. the agent-oriented model) and several important properties of our models.

6.2.1. The Effectiveness of Our Models. In this section, we show the experimental results of the four indices (sr, T, T_c and T_w) by comparing our multiplex network-adapted models (network layer-oriented model and agent-oriented model) with the benchmark models (traditional simplex network-adapted model and transparent model). The effectiveness of our models can be validated from these tests.

Fig. 2(a) reports the results of the total time costs of the four models. The total time cost can reflect the effectiveness of the model to allocate tasks in the system; the total time cost for the same number of tasks negatively correlates with the effectiveness of the model for allocating tasks. This plot indicates that the transparent model has the best performance during the task allocation process; the performance of the network layer-oriented model and agent-oriented model proposed in this paper are close to that of the transparent model; and the traditional simplex network-adapted model performs the worst. Therefore, our models can be validated as effective for task allocation in unreliable multiplex networks.



(a) Total time costs

(b) Success rate

(c) Communication time costs

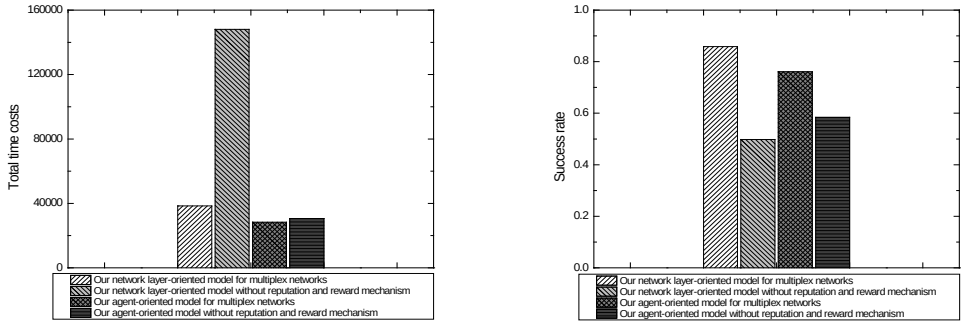
(d) Waiting time costs

Fig. 2. The effectiveness tests

Fig. 2(b), (c), and (d) show the test results of the success rate, the communication time costs, and the waiting time costs, respectively. They can indicate the extent to which the models can satisfy the three objectives for task allocation described in Equation (2) for multiplex networks.

Fig. 2(b) shows that the success rate of our models can nearly continuously increase throughout the entire task allocation processes to finally achieve a good performance level close to that of the transparent model. This trend reveals that the reliability of our model can evolve as the size of tasks increases. In the very early stage (approximately 5 tasks) of task allocation, the success rate of our models shows a nearly 5% decrease. In this stage, the reputation and reward mechanisms of our models are still at the initializing state; after some tasks have been executed, they will play a more effective role. Thus, the success rate can then continuously increase for the following tasks. Additionally, the network layer-oriented model shows a better evolution property compared with the agent-oriented model, which will be analyzed in Section 6.2.3. It notes that the transparent model can detect all the unreliable agents, but its success rate cannot reach 100%; this phenomenon is caused by the network loss rate during resource communication, which may lead to unsuccessful resource access.

Fig. 2(c) and (d) show the test results of the communication time costs and the waiting time costs, respectively. In Fig. 2(c) and (d), the transparent model also performs the best; our models perform much better than the traditional simplex network-adapted model and a little worse than the transparent model. The potential reason



(a) Total time costs (b) Success rate

Fig. 3. The effect of the reputation and reward mechanism in our models

for this difference is that the transparent model can directly allocate the tasks to the truthful agents with a relatively lower communication cost and shorter waiting queues of tasks; thus, it can have the best performance on the two indices. Specifically, the performance of the agent-oriented model is very close to that of the transparent model in terms of the communication time costs; while its performance is closer to that of the network layer-oriented model in terms of the waiting time costs.

The results in the four plots of Fig. 2 indicate **that our models can effectively allocate tasks in unreliable multiplex networks, which can satisfy the objectives of task allocation with load balancing in Equation (2). Their performance can be close to that of the transparent task allocation model and much better than the traditional simplex network-adapted task allocation model.**

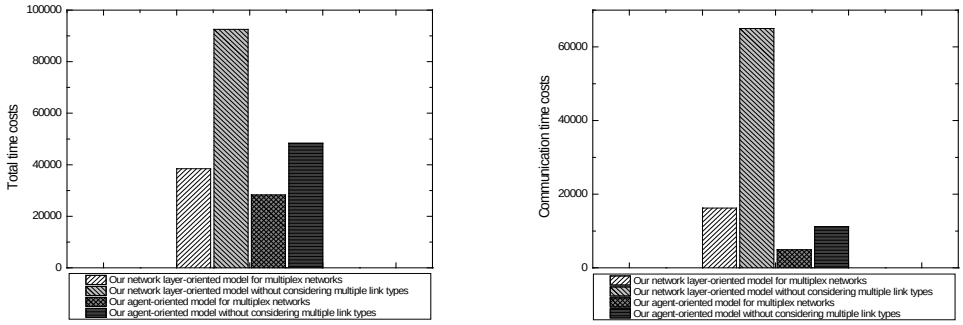
6.2.2. The Effects of the Key Components in Our Models. In this section, we aim to confirm the effects of the following key components in our models: 1) the reputation and reward mechanism; 2) considering multiple link types in resource negotiation; and 3) the contextual load balancing mechanism. To evaluate the effects of these components, we remove each component from our models and then use these modified models to perform the same tests as those used for the original models. By comparing the corresponding test results, the effect of each component can be investigated.

▷ *The Effect of The Reputation and Reward Mechanism*

Fig. 3 shows the test results of the network layer-oriented model, the agent-oriented model, and their modified models without the reputation and reward mechanism. In this test, we concentrate on the indices of the total time costs and the success rate by considering the main effect of the reputation and reward mechanism.

Fig. 3 indicates that the reputation and reward mechanism can not only improve the performance of our models on the success rate but also further improve the performance on the total time costs. This finding proves that **the reputation and reward mechanism positively affects the task allocation in unreliable multiplex networks.**

Moreover, the reputation and reward mechanism plays a more important role in the network layer-oriented model than in the agent-oriented model. This finding can be attributed to the network layer-oriented model, which not only implements the repu-



(a) Total time costs (b) Communication time costs

Fig. 4. The effect of considering multiple link types in resource negotiation in our models

tation and reward mechanism to the selection of agents but also to the allocation of network layers.

▷ *The Effect of Considering Multiple Link Types*

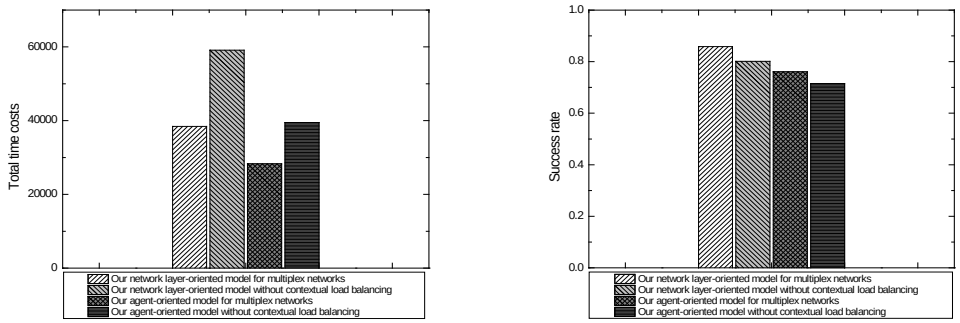
The novelty of our models consists of considering multiple link types in the resource negotiation of agents. Fig. 4 reports the test results of the network layer-oriented model, the agent-oriented model, and their modified models without considering multiple link types in resource negotiation. In this test, we mainly focus on the performance of the total time costs and the communication time costs. The test results of the total time costs shown in Fig. 4(a) indicate that the modified models without the aforementioned consideration perform much worse than the original models; therefore, the consideration of link types in resource negotiation significantly positively affects both our two models. Fig. 4(b) provides a similar conclusion. **In conclusion, our models can effectively address multiple link types in multiplex networks.**

▷ *The Effect of The Contextual Load Balancing Mechanism*

In this paper, we propose the contextual load balancing mechanism in Section 5.4 mainly to decrease the waiting time costs of tasks in multiplex networks. Moreover, implementing contextual load balancing mechanism is expected to positively affect the overall success rate and the communication time costs.

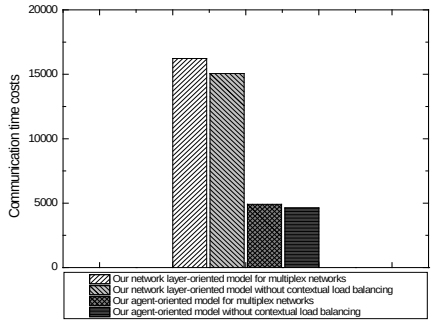
The results in Fig. 5 confirm that the contextual load balancing mechanism reaches the expected aims. From Fig. 5(d), **we can first confirm the positive effect of the contextual load balancing mechanism on decreasing the waiting time costs of tasks.** The waiting time costs of the modified models without the contextual load balancing mechanism are much higher than those of the original models. Fig. 5(a) and (b) indicate **that the total time costs of the modified models are much higher than those of the original models, and the success rate of task allocation is lower.** Finally, Fig. 5(c) shows that the communication time costs only slightly increase after implementing the contextual load balancing mechanism.

6.2.3. Comparison between Our Two Models and Several Key Properties. In this section, we compare our presented network layer-oriented model with the agent-oriented model in multiplex networks. In addition to the four indices (sr, T, T_c, T_w), we also compare the algorithm complexity and the allocation time of the two models. We then test several

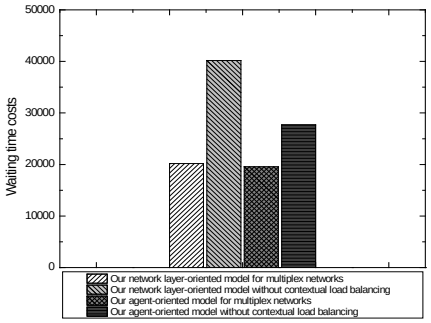


(a) Total time costs

(b) Success rate



(c) Communication time costs



(d) Waiting time costs

Fig. 5. The effect of the contextual load balancing mechanism in our models

Table I. COMPARISON OF THE PROPOSED TWO MODELS FOR MULTIPLEX NETWORKS (NETWORK LAYER-ORIENTED VS. AGENT-ORIENTED)

	Our proposed models	
	Network layer-oriented	Agent-oriented
Allocation algorithm complexity	$O(\|R_t\| \cdot \lambda^2)$	$O(\|R_t\| \cdot \|A\|^2)$
Allocation time	33.45	26462.64
Total time costs (T)	38446.31	28316.86
Success rate (sr)	86%	76%
Communication time costs (T_C)	16225.43	4912.43
Waiting time costs (T_W)	20189.53	19579.20

key properties of our models, such as the robustness for dynamic unreliable environments, the evolution property, and the adaptability to varying situations.

Table 1 shows that the network layer-oriented model outperforms the agent-oriented model in terms of the algorithm complexity, the allocation time, and the success rate; but the agent-oriented model outperforms the network layer-oriented model in terms of the communication time costs. We introduce this result in detail as follows.

Generally, the agent number in the entire network is much larger than the number of network layers; thus, the algorithm complexity of the network layer-oriented model

is much smaller than that of the agent-oriented model. The allocation time reflects the allocation algorithm complexity. Based on the results of the allocation time, we can conclude that the ratio of the allocation time of the network layer-oriented model to that of the agent-oriented model coincides with the ratio of their algorithm complexities.

In addition to the reputation and reward mechanism for the agents, the network layer-oriented model also implements a reputation and reward mechanism for the network layers. As such, a task is allocated through reputation filtering twice in the network-oriented model. Thus, the network layer-oriented model can perform better with respect to the success rate of task allocation.

However, even if the success rate of the network layer-oriented model is higher than that of the agent-oriented model, the agent-oriented model performs better with respect to the total time costs and communication time costs. This difference can primarily be attributed to the agent-oriented model, which can select the path with the lowest communication costs between agents to negotiate resources from the entire network when allocating tasks. Thus, it can significantly reduce the communication time costs for task allocation in multiplex networks.

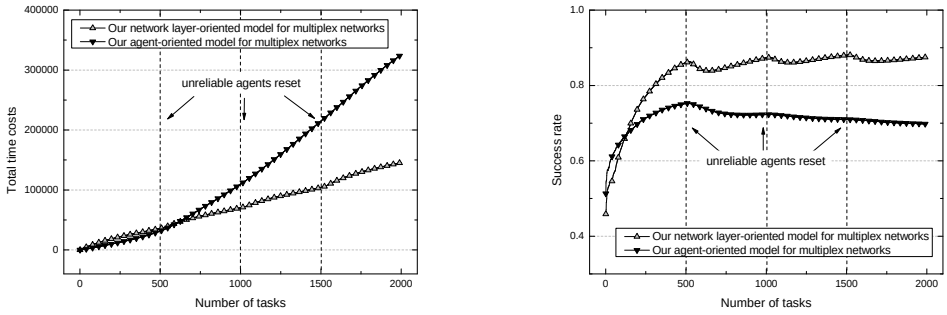
In conclusion, the network layer-oriented model can achieve a higher success rate and significantly lower allocation time compared to the agent-oriented model; the agent-oriented model can achieve lower communication time costs than the network layer-oriented model. Therefore, we can say that the advantage of network layer-oriented model is that it is good at improving the success rate with a significantly lower allocation time, and the advantage of agent-oriented model is that it is good at reducing the communication time costs.

▷ *Robustness for Dynamic Environments*

In this section, we test the robustness of our models in two typical types of dynamic environments: i) the dynamic unreliable environment where agents may dynamically change their identities, and ii) the dynamic environment with unstable services where the resources of agents may be out of service and recovered dynamically. We show the environment settings and the test results as follows.

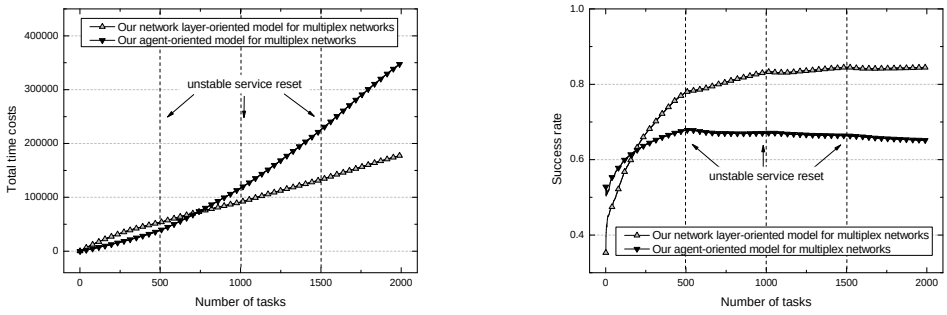
The dynamic unreliable environments in experiments can be set as follows: the unreliable agents in the network will be reset dynamically when the number of tasks allocated is 500, 1000, and 1500; in other words, the previously unreliable agents may become normal after each reset, while some normal agents may become unreliable.

Fig. 6 shows the performance of our models (the network layer-oriented model and the agent-oriented model) in these dynamic unreliable environments. Based on the results of the total time costs and the success rate of the two models, we can first conclude that the network layer-oriented model performs better than the agent-oriented model in the robustness test for the dynamic unreliable environments. When the unreliable agents are dynamically reset, the success rate of task allocation using the network layer-oriented model first decreases, but the success rate soon recovers after the allocation of some tasks. Moreover, the total time costs of the network layer-oriented model increase more rapidly for the first few tasks after the unreliable agents are reset, but the rate of increase can still converge to the former state. We can then analyze the results of the agent-oriented model. The success rate significantly decreases after the first reset of the unreliable agents; however, this decrease can be halted even following the dynamic reset of unreliable agents to finally yield a success rate of nearly 70%. These results indicate the robustness of the agent-oriented model for dynamic



(a) Total time costs (b) Success rate

Fig. 6. The robustness of our models for the dynamic unreliable environment where agents may dynamically change their identities



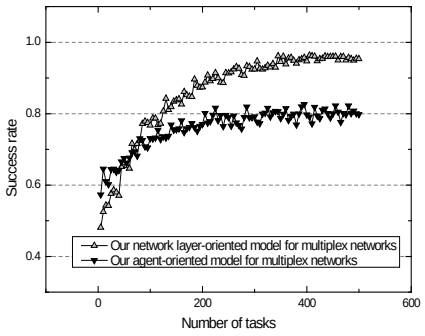
(a) Total time costs (b) Success rate

Fig. 7. The robustness of our models for the dynamic environment with unstable services where the resources of agents may be out of service and recovered dynamically

situations, although the robustness of the network layer-oriented model is better. The results of the total time costs in Fig. 6(a) yield a similar conclusion.

The dynamic environment with unstable services is set as follows: part of resources of agents may be out of service during the task allocation process (this kind of agents is called the *unstable agents*); then after some time, the resources out of service can also be recovered. In this test, besides the unreliable agents in the system, we randomly select additional 10% agents from the system to be the unstable agents. When the number of tasks allocated is 500, 1000, and 1500, the unstable agents will be recovered to be the normal agent and the same number of normal agents selected randomly from the system will be set to be the unstable agent.

Fig. 7 indicates the performance of our models (the network layer-oriented model and the agent-oriented model) in the dynamic environment with unstable services. Fig. 7(a) and (b) show the test results of the total time costs and success rate of our models respectively. First, despite that the total time costs and success rate of our models in this environment are not as good as that no unstable agents are involved in the task allocation process (see Fig. 2), the performance of our models can also be satisfying because there are less available resources for task execution in the system caused by



(a) Success rate of every 5 tasks

Fig. 8. The evolution property of our models during task allocation process

the involved unstable services. Then, we also observe that our models can have a good resistance to the reset of unstable services. After each reset of unstable services when the number of tasks allocated is 500, 1000, and 1500, there are small fluctuations of the success rate and total time costs, i.e., there are no remarkable increase of total time costs and no remarkable decrease of the success rate of our models in this type of dynamic environment. Finally, similar with the conclusion derived from the dynamic unreliable environment, the network layer-oriented model performs better than the agent-oriented model in the dynamic environment with unstable services.

In conclusion, our two models can robustly allocate tasks in dynamic environments where agents may dynamically change their identities and the resources of agents may be out of service and recovered dynamically; and the robustness of the network layer-oriented model is better than that of the agent-oriented model for these environments.

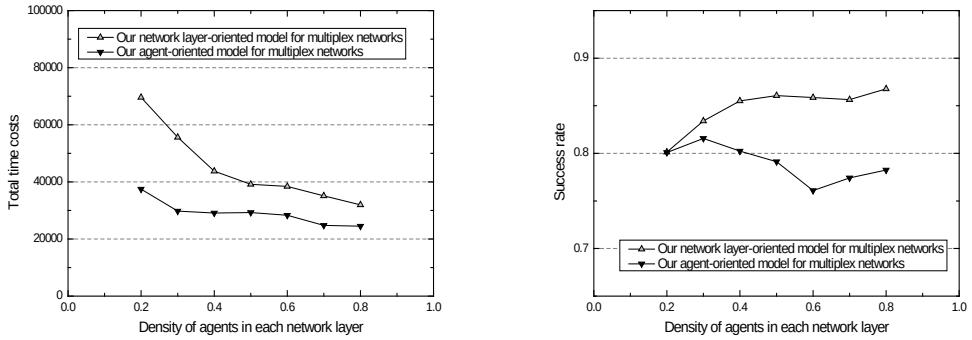
▷ *Evolution Property during Task Allocation Process*

In this section, we use a new index defined as the success rate of every 5 tasks to indicate the evolution property of our models during the task allocation process; this index is different from the success rate, sr, because it does not count the cumulative success rate from the first task to the current task. Fig. 8 indicates that the network layer-oriented model can reach a success rate of nearly 95% after the allocation of 350 tasks. In other words, the reputation and reward mechanism in our model can evolve to largely increase the success rate of task allocation. The agent-oriented model also shows a good evolution property, although this property is not quite as good as that of the network layer-oriented model. The main trend of this test shown in Fig. 8 is very similar to the test results shown in Fig. 2(b).

Based on this result, we can also conclude that the fluctuation of the success rate of every 5 tasks during the task allocation process is always small, i.e. the evolution properties of our models are also quite stable for the task allocation in unreliable multiplex networks, which can result in a very high success rate .

▷ *Adaptability for Varying Agent Densities within Network Layer*

In this section, we test the adaptability of our models for different agent numbers in each network layer. Specifically, this test varies the density of agents in each network



(a) Total time costs (b) Success rate

Fig. 9. The adaptability of our models for varying agent densities within network layer

layer. We mainly focus on the performance of our models with respect to the total time costs and the success rate. Fig. 9 shows the test results.

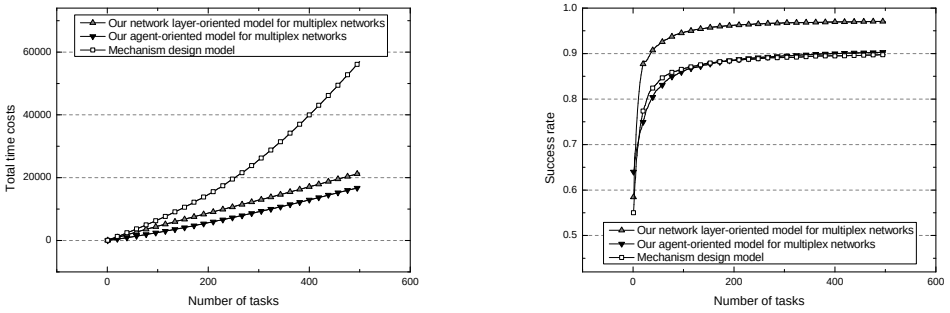
Fig. 9(a) shows that the performance of our models positively correlates with the number of agents in each network layer, and the adaptability of the network layer-oriented model is better than that of the agent-oriented model. This observation can be attributed to the increasing overlap of network layers when the number of agents increases. Thus, more agents act as intermediates for resource negotiation between network layers. In other words, the path for resource negotiation between agents via different network layers can be shorter. Therefore, the communication time costs between agents in task allocation can be reduced.

Fig. 9(b) indicates that the density of agents in each network layer positively correlates with the success rate of the network layer-oriented model and slightly negatively correlates with success rate of the agent-oriented model. This relationship can be attributed to the reputation of agents, which is more significantly influenced by the resource loss rate of network layers they have resided in when the density of agents in each network layer is larger. This effect may lead to larger errors of the evaluation of agent reputations, which actually decreases the reliability of task allocation of the agent-oriented model. However, the network layer-oriented model implements a special reputation and reward mechanism for the network layers, which avoids this problem.

▷ *Applicability for the Environment Considering Task Execution Payoff and Tax*

In some previous benchmark studies (e.g. [Weerdt et al. 2012]), the mechanism design model is often used. In the mechanism design model, each agent may be associated with some utilities; and the final objective of task allocation and execution is to maximize the total utilities of the entire system. Therefore, in the environments of these related studies, each agent should pay a fixed utility of tax when it participates in the task allocation process, and the agent can gain some utilities of payoffs if the tasks participated by the agent can be executed successfully.

Now we will compare our models and the mechanism design model in these environments where the payoff and tax of agents are considered. The mechanism design model encourages unreliable agents to behave reliably by setting proper task execution payoff and tax. The main difference between our models and the mechanism design model is that the mechanism design model does not have specific approaches for managing



(a) Total time costs (b) Success rate

Fig. 10. The applicability of our models for the environment considering task execution payoff and tax

different link types in multiplex networks and does not incorporate any reputation mechanisms.

In the test, only the unreliable agents can be influenced by the new setting of payoff and tax, because the reliable agents are assumed to be always cooperative. Moreover, it is assumed that unreliable agent also expect to gain benefit during the task allocation process. The manner of an unreliable agent a_i for executing tasks can be determined by two aspects that are i) the total payoffs it has gained, P_i, and ii) the total taxes it has paid, Tax_i. When $Tax_i > P_i$, a_i will perform as a reliable agent in the future task execution; and when $Tax_i < P_i$, it will perform unreliably.

Fig. 10(a) shows the test results of the three models on the index of total time costs. Fig. 10(a) indicates that our models can perform much better than the mechanism design model if there are more tasks being allocated and executed by the system. The main reason is that the mechanism design model cannot manage the different link types in task allocation, i.e., different relative biases for communicating different types of resources. Hence, the communication time costs of the mechanism design model should be much larger than our models. Moreover, our models can also have higher success rate than the mechanism design model (see Fig. 10(b)); thus the time costs for reallocation of failed tasks can also be reduced.

Fig. 10(b) shows the test results of the three models on the index of success rate. From Fig. 10(b), we find that all three models can have a satisfying performance in success rate; the agent-oriented model can perform closely to the mechanism design model; and the network layer-oriented model can have a much better performance than the other two models.

Therefore, from the above experimental results, we can conclude that our models can adapt to the environments where the payoff and tax of agents are considered.

7. CONCLUSIONS

Generally, the allocation of tasks in unreliable NMASs has three objectives [Jiang et al. 2013b]: 1) improve the reliability with which tasks will obtain true resources for successful execution; 2) reduce the communication time of tasks to access required resources in networks; and 3) reduce the waiting time of tasks for the required resources. These objectives have already been achieved for simplex networks in previous benchmark works. However, new problems emerge when we implement these three objectives in multiplex networks that contain multiple types of links between agents,

where each type of link may have different relative biases and reliability in communicating different type of resources.

To solve the new problems, we consider both the link types and the agents and substantially extend the existing work by highlighting the effect of network layers on the task allocation and load balancing. Moreover, we present a novel task allocation target, the network layer, which is composed of the same type of links and involved agents; we then propose two models of reliable task allocation with load balancing for multiplex networks, the network layer-oriented model and agent-oriented model, both of which consider the link characteristics of multiplex networks. In the network layer-oriented model, the network layers that can satisfy the objectives of task allocation will first be allocated, and the final agents will then be selected from the allocated network layers; in the agent-oriented model, the allocated agents are directly selected from all of the agents in the entire network. The latter model can optimize communication to save time but incur excessive allocation times; the former can improve the reliability and significantly reduce the task allocation time, although it may slightly increase the communication time. When there are a set of hybrid tasks, we can combine these two models into the task allocation as the following: the agent-oriented model can be used for the tasks that may produce much communication time while the network layer-oriented model can be used for the tasks that may be highly critical and need higher reliability.

The theoretical analyses and experimental results prove that our two presented models can effectively satisfy the task allocation objectives in unreliable multiplex networks and significantly reduce the time costs and improve the success rate of tasks for multiplex networks over the traditional simplex network-adapted task allocation model. Moreover, our models show a favorable robustness, evolution property, and adaptation property for varying real situations.

In this paper, the network layers are assumed to only differ essentially in their communication biases on different types of resources. However, in some circumstances, agents may have different computing capabilities, such as CPU power and memory storage, thus the network layers may also differ in computing capabilities. To address this problem, we can consider two situations in the future, shown as the followings.

(1) If the computing units can migrate among agents in the network, we only need to abstract the computing capabilities to resources and still use our presented model to make task allocation and load balancing.
(2) If the computing units cannot migrate, we should let the tasks migrate among agents in network. Therefore, it needs to make minor revisions for our presented model: the network layers' communication biases for tasks should be addressed. Certainly, the main frameworks and algorithms of the model can still be used in this case.

Therefore, our presented model can easily be extended into other situations in which different network layers have different computing capabilities.

REFERENCES

Abdallah, S. and Lesser, V. 2007. Multiagent Reinforcement Learning and Self-Organization in A Network of Agents, In *Proceedings of the 6th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'07)*, Honolulu, Hawaii, 172-179.

An, B., Lesser, V. and Sim, K. M. 2011. Strategic Agents for Multi-Resource Negotiation, *Journal of Autonomous Agents and Multi-Agent Systems*, 23,1, 114-153.

Aknine, S., Pinson, Suzanne and Shakun, Melvin F. 2004. An Extended Multi-Agent Negotiation Protocol, *Journal of Autonomous Agents and Multi-Agent Systems*, 8,1, 5-45.

Boccaletti, S., Bianconi, G., Criado, R., Genio, C. I. del, Gómez-Gardeñes, J., Romance, M., Sendiña-Nadal, I., Wangk, Z. and Zaninm, M. 2014. The Structure and Dynamics of Multilayer Networks, *Physics Reports*, DOI: 10.1016/j.physrep.2014.07.001.

Brummitt, C.D., Lee, K.M. and Goh, K.I. 2012. Multiplexity-Facilitated Cascades in Networks, *Physical Review E*, Vol. 85, 045102(R).

Buldyrev, S. V., Parshani, R., Paul, G., Stanley, H. E. and Havlin, S. 2010. Catastrophic cascade of failures in interdependent networks, *Nature*, 464, 7291, 1025-1028.

Chow, K.P. and Kwok, Y.K. 2002. On Load Balancing for Distributed Multiagent Computing, *IEEE Transactions on Parallel and Distributed Systems*, 13,8, 787-801.

Cozzo, E., Baños, R. A., Meloni, S. and Moreno, Y. 2013. Contact-based social contagion in multiplex networks. *Physical Review E*, 88, 5, 050801.

Das, A. and Islam, M. M. 2012. SecuredTrust: A Dynamic Trust Computation Model for Secured Communication in Multiagent Systems, *IEEE Transactions on Dependable and Secure Computing*, 9,2, 261-274.

Fjuita, S. and Lesser, V. R. 1996. Centralized Task Distribution in The Presence of Uncertainty and Time Deadlines, In *Proceedings of the Second International Conference on Multiagent Systems (ICMAS'96)*, Kyoto, Japan,87-94.

Gómez-Gardeñes, J., Reinares, I., Arenas, A. and Floría, L.M. 2012. Evolution of Cooperation in Multiplex Networks, *Scientific Reports*, 2,620.

Gómez, S., Díaz-Guilera, A., Gómez-Gardeñes, J., Pérez-Vicente, C. J., Moreno, Y. and Arenas, A. 2013. Diffusion Dynamics on Multiplex Networks, *Physical Review Letters*, 110, 2, 028701.

Hong, B. and Prasanna, V.K. 2007. Adaptive Allocation of Independent Tasks to Maximize Throughput, *IEEE Transactions on Parallel and Distributed Systems*, 18,10, 1420-1435.

Jiang, Y. and Jiang, J. 2009. Contextual Resource Negotiation-Based Task Allocation and Load Balancing in Complex Software Systems, *IEEE Transactions on Parallel and Distributed Systems*, 20,5, 641-653.

Jiang, Y. and Huang, Z. 2012. The Rich Get Richer: Preferential Attachment in the Task Allocation of Cooperative Networked Multiagent Systems with Resource Caching, *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*,42,5, 1040-1052.

Jiang, Y., Zhou, Y. and Li, Y. 2013a. Network Layer-Oriented Task Allocation for Multiagent Systems in Undependable Multiplex Networks, In *Proceedings of the 2013 IEEE International Conference on Tools with Artificial Intelligence (ICTAI'13)*, Washington, DC, USA, November 4-6.

Jiang, Y., Zhou, Y. and Wang, W. 2013b. Task Allocation for Undependable Multiagent Systems in Social Networks, *IEEE Transactions on Parallel and Distributed Systems*, 24,8, 1671-1681.

Jiang, Y. and Jiang, J. C. 2014. Understanding Social Networks from a Multiagent Perspective, *IEEE Transactions on Parallel and Distributed Systems*, 25,10, 2743-2759.

Kota, R., Gibbins, N. and Jennings, N. R. 2012. Decentralised Approaches for Self-Adaptation in Agent Organizations, *ACM Transactions on Autonomous and Adaptive Systems*, 7,1, 1-28.

Kraus, S. and Plotkin, T. 2000. Algorithms of Distributed Task Allocation for Cooperative Agents, *Theoretical Computer Science*, 242,1-2, 1-27.

Lee, K. M., Kim, J. Y., Cho, W. K., Goh, K. I. and Kim, I. M. 2012. Correlated Multiplexity and Connectivity of Multiplex Random Networks, *New Journal of Physics*, 14, 033027.

Li, Z. and Jiang, Y. 2014. Cross-Layers Cascade in Multiplex Networks, *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS-14)*, Paris, France, May 5-9, 269-276.

Liu, J.,Jin, X. and Wang, Y. 2005. Agent-based Load Balancing on Homogeneous Minigrids: Macroscopic Modeling and Characterization, *IEEE Transactions on Parallel and Distributed Systems*, 16,7, 586-598.

Ma, P. R., Lee, E. Y. S. and Tsuchiya, M. 1982. A Task Allocation Model for Distributed Computing Systems, *IEEE Transactions on Computers*, C-31,1, 41-47.

Ohtsuki, H., Hauert, C., Lieberman, E. and Nowak, M.A. 2006. A Simple Rule for The Evolution of Cooperation on Graphs and Social Networks, *Nature*, vol.441, 502-505.

Salehi, M., Sharma, R., Marzolla, M., Montesi, D., Siyari, P. and Magnani, M. 2014.Diffusion Processes on Multilayer Networks, *arXiv:1405.4329 [cs.SI]*, http://arxiv.org/abs/1405.4329.

Shatz, S.M., Wang, J.P. and Goto, M. 1992. Task Allocation for Maximizing Reliability of Distributed Computer Systems, *IEEE Transactions on Computers*, 41,9, 1156-1168.

Shehory, O. and Kraus, S. 1988. Methods for Task Allocation via Agent Coalition Formation, *Artificial Intelligence*, 101,1-2, 165-200.

Smith, R. G. 1980. The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver, *IEEE Transactions on Computers*, 29,12, 1104-1113.

Szell, M., Lambiotte, R. and Thurner, S. 2010. Multirelational Organization of Large-Scale Social Networks in an Online World, In *Proceedings of the National Academy of Sciences of the United States of America*, 107,31, 13636-13641.

Wang, Z., Szolnoki, A. and Perc, M. 2013. Optimal Interdependence between Networks for the Evolution of Cooperation, *Scientific Reports*, 3, 2470.

Wang, Z., Wang, L. and Perc M. 2014. Degree Mixing in Multilayer Networks Impedes the Evolution of Cooperation, *Physical Review E*, 89, 052813.

Weerdt, M. M., Zhang, Y. and Klos, T. 2012. Multiagent Task Allocation in Social Networks, *Autonomous Agents and Multi-Agent Systems*, 25,1, 46-86.

Xu, J., Lam, A. Y. S. and Li, V. O. K. 2011. Chemical Reaction Optimization for Task Scheduling in Grid Computing, *IEEE Transactions on Parallel and Distributed Systems*, 22,10, 1624-1631.

Yağan, O. and Gligor, V. 2012. Analysis of Complex Contagions in Random Multiplex Networks, *Physical Review E*, Vol.86,036103.

Ye, D., Zhang, M. and Sutanto, D. 2013. Self-Adaptation Based Dynamic Coalition Formation in A Distributed Agent Network: A Mechanism and A Brief Survey, *IEEE Transactions on Parallel and Distributed Systems*, 24,5, 1042-1051.

Zhang, K., Jr., E. G. C. and Shi, D. 2012. Centralized and Distributed Task Allocation in Multi-Robot Teams via A Stochastic Clustering Auction, *ACM Transactions on Autonomous and Adaptive Systems*, 7, 2, article 21.

Zhou, X., Ippoliti, D. and Boult, T. 2007. Hop-Count Based Probabilistic Packet Dropping: Congestion Mitigation with Loss Rate Differentiation, *Computer Communications*, 30,18, 3859-3869.

Zlotkin, G. and Rosenschein, J. S. 1991. Incomplete Information and Deception in Multi-Agent Negotiation, In *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI'91)*, Sydney, Australia, 225-231.

Yichuan Jiang 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 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 IEEE Transactions on Systems, Man, and Cybernetics: Systems, the IEEE Transactions on Cybernetics, 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). He won the best paper award and best student paper award, respectively, from PRIMA and ICTAI. He is a member of ACM, a senior member of IEEE, a member of 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 Chinese Journal of Computers.