Contextual Resource Negotiation-Based Task Allocation and Load Balancing in Complex Software Systems

Yichuan Jiang, *Member*, *IEEE*, and Jiuchuan Jiang

Abstract—In the complex software systems, software agents always need to negotiate with other agents within their physical and social contexts when they execute tasks. Obviously, the capacity of a software agent to execute tasks is determined by not only itself but also its contextual agents; thus, the number of tasks allocated on an agent should be directly proportional to its self-owned resources as well as its contextual agents' resources. This paper presents a novel *task allocation* model based on the *contextual resource negotiation*. In the presented task allocation model, while a task comes to the software system, it is first assigned to a principal agent that has high contextual enrichment factor for the required resources; then, the principal agent will negotiate with its contextual agents to execute the assigned task. However, while multiple tasks come to the software system, it is necessary to make load balancing to avoid overconvergence of tasks at certain agents that are rich of contextual resources. Thus, this paper also presents a novel *load balancing* method: if there are overlarge number of tasks queued for a certain agent, the capacities of both the agent itself and its contextual agents to accept new tasks will be reduced. Therefore, in this paper, the task allocation and load balancing are implemented according to the *contextual resource distribution* of agents, which can be well suited for the characteristics of complex software systems; and the presented model can reduce more communication costs between allocated agents than the previous methods based on *self-owned resource distribution* of agents.

Index Terms—Complex software systems, multiagents, contextual resource, negotiation, task allocation, load balancing.

1 Introduction

Now, many software systems are very complex and have a large number of parts that interact with each other [1]; to manage the complexity of those complex software systems, agent-oriented approaches are well suited for modeling and developing them [2], [3], [4], [5]. With the agent-oriented approaches, the systems always contain many software agents that have many interactions under some organizational constraints and work together to achieve certain objectives occurring through the systems. When a task comes to a complex software system, the software agents in the system should coordinate their activities to implement the task; therefore, coordination among those software agents, which mainly includes two aspects: *task allocation* and *load balancing*, is considered as one of the key concepts to implement tasks effectively in a real complex system [6], [7], [8], [9], [27].

When a multiagent complex system wants to execute a task, the first step is to allocate the task to some software agents, which is called task allocation. The main goal of task allocation is to maximize the overall performance of the system and to fulfill the tasks as soon as possible [6], [7], [8]. Generally, the task allocation in the previous related work is always implemented based on the agents' self-owned resource distribution; the number of allocated tasks on an agent is always directly proportional to its *self-owned resources*, i.e., an agent may be assigned with more tasks if it owns more resources by itself [8], [9], [10], [11].

For example, if agent a_1 holds the resource set $\{r_1, r_2\}$, a_2 holds the resource set $\{r_1\}$, and a task t needs the resource set $\{r_1, r_2, r_3\}$. According to the previous task allocation method based on self-owned resource distribution, a_1 may have higher probability to obtain task t. However, an agent may share resources by negotiating with other agents in the system; now, it is assumed that a_2 can easily share resources $\{r_2, r_3\}$ with its interacting counterparts, but a_1 cannot share any resources from other agents, thus we should allocate task t to agent a_2 but not a_1. Therefore, in this paper, we present a new idea of task allocation: *if an agent does not own plentiful resources by itself, but it can obtain enough resources from other interacting agents easily, it may also be allocated tasks; the number of allocated tasks on an agent is directly proportional to not only its own resources but also the resources of its interacting agents*.

The complex software systems are always modeled both from physical and social viewpoints. Generally, each agent locates at a physical place and has some physically interacting agents in the complex system; on the other hand, the complex systems are built on the socially organized multiagents, thus each software agent also locates in some social organizations [12], [13], [14], [15]. Therefore, each agent in the complex system may have two kinds of contexts, one is the physical context, and the other is the social context [16], [17], [18]. The resources owned by the agents within the contexts of an agent are called contextual resources. Obviously, agents will often negotiate

- *Y. Jiang is with the Research Center for Learning Science, Southeast University, Nanjing 210096, P.R. China and the State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100080, P.R. China. E-mail: yjiang@seu.edu.cn, jiangyichuan@yahoo.com.cn.*
- *J. Jiang is with the School of Information Science and Engineering, Southeast University, Nanjing 210096, P.R. China. E-mail: jcjiang@163.com.*

Manuscript received 20 June 2007; revised 1 Dec. 2007; accepted 25 June 2008; published online 21 July 2008.
Recommended for acceptance by R. Eigenmann.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2007-06-0200.
Digital Object Identifier no. 10.1109/TPDS.2008.133.

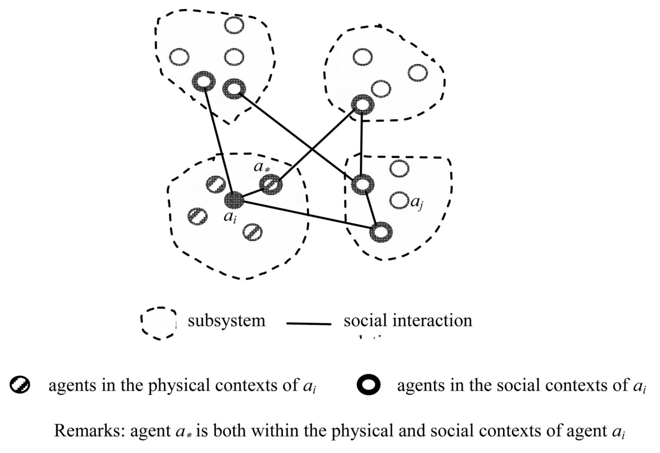

a_*
a_i
a_j
subsystem
social interaction
agents in the physical contexts of a_i
agents in the social contexts of a_i
Remarks: agent a_* is both within the physical and social contexts of agent a_i

Fig. 1. An example to demonstrate the physical and social contexts in the complex system.

with the agents within their physical and social contexts when they execute tasks. Therefore, our idea of the task allocation in complex software systems can also be explained as follows: *the number of allocated tasks on an agent is directly proportional to not only its self-owned resources but also the resources of its contextual agents; the task allocation of system should be determined by the contextual resources distribution of agents*.

Fig. 1 shows those two kinds of contexts of agents in a complex system, where the subsystem is considered as the physical context and the set of organizationally interacting agents is considered as the social context. We let the set of resources owned by a_i be $\{r_1, r_2\}$ and the set of resources owned by a_j be $\{r_1, r_2, r_3\}$. Now, it is assumed that a task needs the resources $\{r_1, r_2, r_3, r_4\}$; if we make task allocation based on self-owned resources, a_j will have higher probability to obtain the task since it can satisfy the resource requirement of the task better than a_i. However, now a_* is within the context of a_i and has the resources $\{r_3, r_4\}$; thus, we can suggest that a_i has higher probability to obtain the task since a_i can obtain the resources easily by negotiating with a_*.

As said above, if an agent possesses more contextual resources, it may be assigned more tasks accordingly. However, if too many tasks are crowded on certain agents that are rich of contextual resources, then the tasks may be delayed and do not obtain quick responses; therefore, we should let some tasks be switched to other agents with relatively fewer contextual resources but lower task loads, which is called *load balancing*. In the related work about load balancing [10], [11], [19], [20], [21], [22], the number of tasks queuing for an agent is the determinative factor for such agent's right in the future task allocation; if there are too many tasks queuing for an agent, the probability of such agent to obtain new tasks will be reduced.

Since every agent is located within some contexts and will negotiate with its contextual agents to borrow resources for executing tasks, therefore, in this paper, we will develop the load balancing method also based on the contextual resource distribution: *if an agent, **a**, is allocated with too many tasks, then it is not only **a** itself but also the agents in the context of **a** should be reduced the possibilities to accept new tasks in the future*.

The main novelty of this paper is that it presents a novel model for task allocation and load balancing based on the contextual resource negotiation but not the self-owned resource distribution among software agents; therefore, our model can be well suited for the characteristics of complex software systems. The rest of this paper is organized as follows: In Section 2, we describe the contextual resource negotiation in complex software systems; in Section 3, we present the task allocation model to meet the contextual resource distribution of complex software systems; in Section 4, we address the context-associated load balancing for multiple tasks; in Section 5, we provide the simulation results to validate our proposed model; in Section 6, we present some discussions on the extension of our model. Finally, we conclude this paper in Section 7.

2 Contextual Resource Negotiation

The context of an agent can be simply regarded as the environment it is situated [26], which includes the physical context and the social context (organizational one). The physical context is produced by the agent's physical environment, which can be regarded as the agent's physical location, and the physically nearby agents within the subsystem; the resources owned by the agents within its physical context are called the *physically contextual resources*. On the other hand, agents in the complex system should be organized within some social organizations [12], so the counterpart agents in the social organizations can be regarded as the agent's social context, and the resources of the agents in the social context are called *socially contextual resource*.

2.1 Physically Contextual Resource

If an agent lacks the necessary resources to implement the allocated task (we call such agent as *initiator agent*), it may negotiate with its physically contextual agents; if the physically contextual agents have the required resources (we call those agents that lend resources to the initiator agent as *response agents*), then the initiator agent and the response agents will cooperate together to implement such task.

Then, which ones will the initiator agent negotiate with? To minimize the negotiation communication costs, the initiator agent can implement the negotiation process from near agents to far agents gradually within the physical environment.

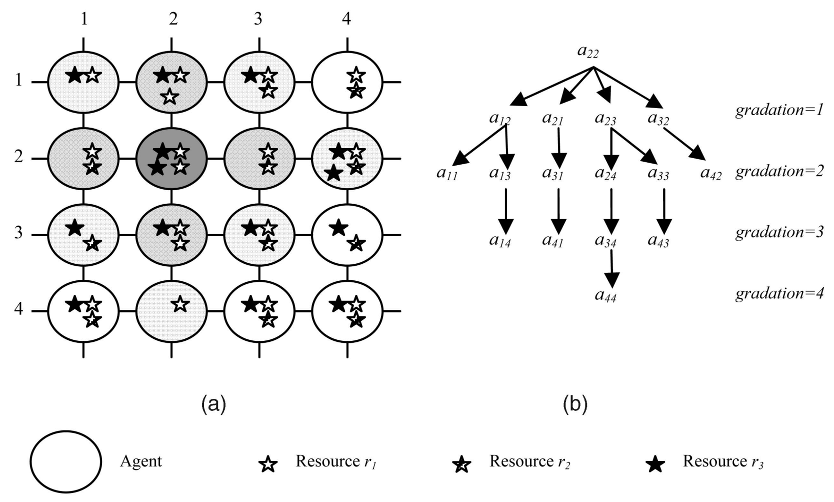
The negotiation relations from agent a to other agents within its physical context form a directed acyclic graph with single source a, which is called the physically contextual resource negotiation topology (PCR-NT) of agent a.

Definition 1. PCR-NT. *A* PCR-NT *of agent a is a directed acyclic graph with single source a, the agents in the graph are the ones negotiated by agent a, and the path length from a to any other agent is the physical negotiation gradation of that agent.*

Definition 2. Physical negotiation gradation. *Let a be an agent that will negotiate with other agents within its physical context and the agents in the nth round of negotiation of agent a be called the physical contexts with gradation n.*

Let a_i be the initiator agent, and the resources owned by a_i be R_{ai}; now, task t is allocated to a_i, and the set of requested resources for implementing t is R_t. Therefore, the set of lacking resources of a_i to implement t is $\overline{R^t_{a_i}}$:

$$\overline{R^t_{a_i}} = R_t - R_{a_i}. \quad (1)$$



1
2
3
4
(a)
a_{22}
a_{12} a_{21} a_{23} a_{32}
gradation=1
a_{11} a_{13} a_{31} a_{24} a_{33} a_{42}
gradation=2
a_{14} a_{41} a_{34} a_{43}
gradation=3
a_{44}
gradation=4
(b)
Agent
Resource r_1
Resource r_2
Resource r_3

Fig. 2. An example for constructing the PCR-NT.

Now, it is assumed that agent a_j is negotiated by a_i, the set of resources owned by a_j is R_{aj}. If a_j has any resources that are requested by a_i to implement t, then the set of resources that a_j can lend to a_i for implementing task t is

$$R^t_{a_j\rightarrow a_i}=\left\{r|r\in R_{a_j}\wedge r\in\overline{R^t_{a_i}}\right\}. \tag{2}$$

Thus, the set of lacking resources of a_i to implement t will be reduced as

$$\overline{R^t_{a_i}}=\overline{R^t_{a_i}}-R^t_{a_j\rightarrow a_i}=\overline{R^t_{a_i}}-R_{a_j}. \tag{3}$$

The initiator agent a_i will negotiate with the physically contextual agents according to the *PCR-NT*, until all requested resources are satisfied. The negotiation process is shown as Algorithm 1.

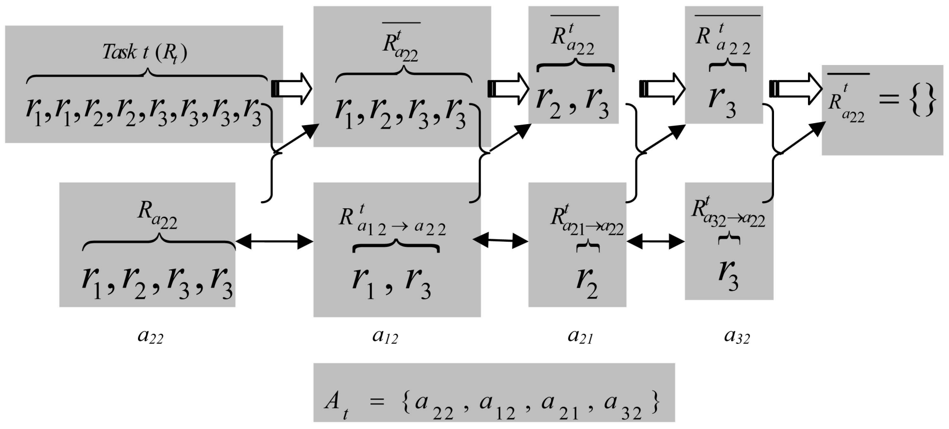
Algorithm 1. Physically contextual resource negotiation of agent a:

```
//$a$ is the initiator agent, $A$ is the physical context of $a$//
1) **Set** the tags for all agents in $A$ to 0 initially;
2) **Create** Queue($Q$);
3) **Insert** Queue ($Q, a$);
4) **Set** the tag of $a$ to 1;
5) $b = 0$;
6) $A_t = \{a\}$;  /*The allocated agent set for task $t$*/
7) $\overline{R^t_a} = R_t - R_a$;  /*The lacking resources of agent $a$ to
   implement task $t$*/
8) **If** $\overline{R^t_a} == \{\}$, **then** $b = 1$;   /*Agent $a$ can provide all
   resources to implement task $t$*/
9) **While** ((!EmptyQueue ($Q$)) and ($b == 0$)) **do**:
  9.1) $aout$ = **Out** Queue($Q$);
  9.2) $R' = \overline{R^t_a} - R_{aout}$;
  9.3) **If** $R' \neq \overline{R^t_a}$, **then**: /*Agent $aout$ can satisfy some
       requests of $a$*/
    9.3.1) $\overline{R^t_a} = \overline{R^t_a} - R_{aout}$; /*Agent $a$ obtains resources
           from $aout$ to implement $t$*/
    9.3.2) $A_t = A_t \cup \{aout\}$;
  9.4) **If** $\overline{R^t_a} == \{\}$, **then** $b = 1$; /*All resources for
       implementing $t$ are satisfied*/
  9.5) **For** $\forall alocal \in L_{aout}$: /*$L_{aout}$ is the set of all physical
       neighbors of $aout$*/
       **if** the tag of $alocal$ is 0, **then**: /*If agent $alocal$ was
       not negotiated by $a$ before*/
    9.5.1) **Insert** Queue ($Q, alocal$);
    9.5.2) **Set** the tag of $alocal$ to 1;
10) **If** ($b == 1$), **then Return** ($A_t$) /*All resources for
    implementing $t$ are satisfied*/
    **else Return** (False);
11) **End**.
```

Theorem 1. *If all resources for implementing task t can be satisfied by using Algorithm 1, the total communication cost between the initiator agent and the physically response agents is the minimum.*

Proof. Let a_i be the initiator agent and the set of lacking resources of a_i to implement t be $\overline{R^t_{a_i}}$. If Algorithm 1 is used, the set of response agents is A_* ($A_* = A_t - a_i$), and the total communication cost between a_i and A_* is C_*. Now, if there is a set of agents A'_*, $A'_* \neq A_*$, which can provide $\overline{R^t_{a_i}}$, and the total communication cost between a_i and A'_* is C'_*; if $C'_* < C_*$, it denotes that there are any agents with higher gradations that provide the required resources in $\overline{R^t_{a_i}}$, but the lower gradation agents with required resources do not provide the required resources in $\overline{R^t_{a_i}}$. Obviously, such situation cannot take place in Algorithm 1. Therefore, we have Theorem 1. □

Example 1. Fig. 2 is an example for constructing the *PCR-NT*, where a_{22} is the initiator agent. In Fig. 2, the set of resources owned by agent a_{22} is $\{r_1, r_2, r_3, r_3\}$. Now, we let a task t be allocated to a_{22}, and the resources requested by t be $\{r_1, r_1, r_2, r_2, r_3, r_3, r_3, r_3\}$; obviously, a_{22} lacks the resources $\{r_1, r_2, r_3, r_3\}$. At first, a_{22} negotiates with a_{12} and obtains the requested resources $\{r_1, r_3\}$; second, a_{22} negotiates with a_{21} and obtains the requested resources $\{r_2\}$; third, a_{22} negotiates with a_{23} but obtains no requested resources; at last, a_{22} negotiates with a_{32} and obtains the requested resources $\{r_3\}$; now, the requested resources of a_{22} to implement t are all satisfied. Therefore, the agents $\{a_{22}, a_{12}, a_{21}, a_{32}\}$ will implement task t corporately. The negotiation process is shown in Fig. 3.



Task t (R_t)
$r_1, r_1, r_2, r_2, r_3, r_3, r_3, r_3$
$\overline{R^t_{a_{22}}}$
r_1, r_2, r_3, r_3
r_2, r_3
r_3
$\overline{R^t_{a_{22}}} = \{\}$
$R_{a_{22}}$
r_1, r_2, r_3, r_3
$R^t_{a_{12} \to a_{22}}$
r_1, r_3
$R^t_{a_{21} \to a_{22}}$
r_2
$R^t_{a_{32} \to a_{22}}$
r_3
a_{22}
a_{12}
a_{21}
a_{32}
$A_t = \{a_{22}, a_{12}, a_{21}, a_{32}\}$

Fig. 3. Physically contextual resource negotiation process of the example in Fig. 2.

2.2 Socially Contextual Resource

The complex systems can be modeled as the multiagents organized within certain social structures [14], [23]. With the social organizations, the social interaction relations describe the obligations and structures among agents in the complex systems [13]. Agents will always interact and negotiate with other agents within the social contexts while they execute tasks.

In the social organizations, it is more likely that the near individuals may have more similarities and, being closer together in the organizational hierarchy, share more common interests than the remote individuals [24]. Therefore, in the social organizations of complex systems, each agent will negotiate with other agents for the requested resources gradually from near places to remote places.

The hierarchical structure is a typical social organization, thus first we address the negotiation of social contextual resources within hierarchical structures; after that, we will address the one within arbitrary social structures.

In the social contexts, the agents negotiate with other agents also according to certain topologies. Referring to Definitions 1 and 2, we now give the definitions of social negotiation gradation and socially contextual resource negotiation topology (SCR-NT).

Definition 3. Social negotiation gradation. *Let a be an agent that will negotiate with other agents within its social context and the agents in the nth round of negotiation of agent a be called the social contexts with gradation n.*

Definition 4. SCR-NT. *An* SCR-NT *of agent a is a directed acyclic graph with single source a, the agents in the graph are the ones negotiated by agent a, and the path length from a to any other agent is the social negotiation gradation of that agent.*

2.2.1 Hierarchical Structure

In the hierarchical structures, each agent can interact *directly* only to its superiors and subordinates; thus, each agent will first negotiate with its superiors or subordinates for resources. Moreover, in the hierarchical organizations, resource negotiation always happens between pairs of agents that share the same immediate superior; and agents will always negotiate resources through the lowest common ancestor [25]. Therefore, let there be an agent a that can negotiate with other agents within the hierarchical structure according to the following orders:

1. the subordinates of agent a in the hierarchical structure;
2. the immediate superior of agent a;
3. the sibling agents with the lowest common superiors.

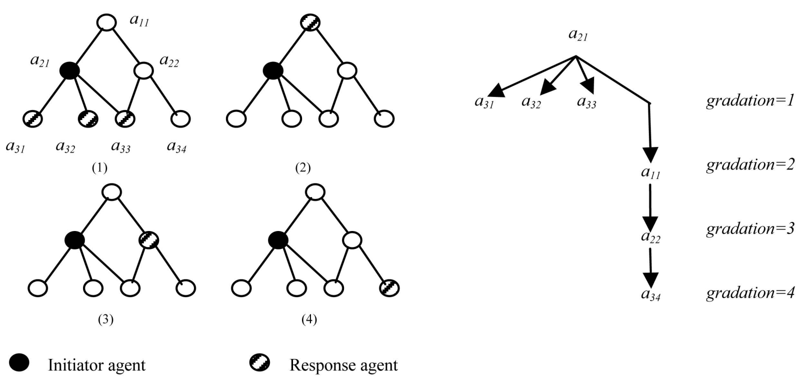
Example 2. In Fig. 4, let agent a_{21} be the initiator agent, we can see the construction of an *SCR-NT* according to the above social negotiation orders in the hierarchical structure.

Let a be the initiator agent, and the set of agents in a's social context be A, now the socially contextual resource negotiation process of agent a in hierarchical structures is shown as Algorithm 2.

Algorithm 2. Socially contextual resource negotiation of agent a in a *hierarchical social structure*.

```
/* $T_x$: the subtree whose root is agent $x$ in the hierarchical
structure; $p_x$: the parent node of $x$ in the hierarchical
structure. */
1) **Set** the tags for all agents in $A$ to 0 initially;
2) $b = 0$;
3) $A_t = \{a\}$;   /*The allocated agent set for task $t$*/
4) $\overline{R^t_a} = R_t - R_a$;   /*The lacking resources of agent $a$ to
   implement task $t$*/
5) **If** $\overline{R^t_a} == \{\}$, **then** $b = 1$;   /*Agent $a$ can provide all
   resources to implement task $t$*/
6) **If** $(b == 0)$, **then**:
  6.1) **Negotiation** $(a, a)$;   /*$a$ negotiates with the agents
       within $T_a$ by calling Algorithm 3*/
  6.2) $atemp = a$;
  6.3) **While** $((b == 0)$ and $(p_{atemp} <> Nil))$ **do**:
    6.3.1) $atemp = p_{atemp}$;
    6.3.2) **Negotiation** $(a, atemp)$;   /*$a$ negotiates with the
           agents within $T_{atemp}$ by calling Algorithm 3*/
7) **If** $(b == 1)$, **then Return** $(A_t)$   /*All resources for
   implementing $t$ are satisfied*/
     **else Return** (False);
8) **End**.
```

We can use the lay-based traversal method in tree to implement the negotiation process in Algorithm 2. The method is shown as Algorithm 3.



	r_1	r_2	r_3
a_{11}	2	0	1
a_{21}	2	1	2
a_{22}	1	2	1
a_{31}	1	0	0
a_{32}	2	1	0
a_{33}	2	0	0
a_{34}	1	2	1

(a) (b) (c)

Fig. 4. An example for constructing the *SCR-NT* in a hierarchical social structure; now, it is assumed that a_{21} is the initiator agent. (a) Negotiation topology evolution. (b) *SCR-NT* of a_{21}. (c) The number of resources owned by agents.

```
Algorithm 3. Negotiation $(a, x)$. /* Agent $a$ negotiates with the agents in subtree $T_x$ */
  1) Create Queue $(Q)$;
  2) Insert Queue $(Q, x)$;
  3) Set the tag of $x$ to 1;
  4) While (!EmptyQueue$(Q)$ and $(b == 0)$) do:
    4.1) $aout$ = Out Queue$(Q)$;
    4.2) $R' = \overline{R_a^t} - R_{aout}$;
    4.3) If $R' \neq \overline{R_a^t}$, then:
      4.3.1) $\overline{R_a^t} = \overline{R_a^t} - R_{aout}$;    /*Agent $a$ obtains resources
             from $aout$ to implement $t$*/
      4.3.2) $A_t = A_t \cup \{aout\}$;
    4.4) If $\overline{R_a^t} == \{\}$, then $b = 1$;    /*All resources for
         implementing $t$ are satisfied*/
    4.5) For $\forall achild \in child(aout)$:
         If the tag of $achild$ is 0:
           4.5.1) Insert Queue$(Q, achild)$;
           4.5.2) Set the tag of $achild$ to 1;
  5) Return $(b)$;
  6) End.
```

Obviously, with Algorithms 2 and 3, we can constrain the resource negotiation process in hierarchical social structures as follows:

Lemma 1. *If all resources for implementing task t can be satisfied by using Algorithms 2 and 3, the set of socially response agents in hierarchical social structure can satisfy one of the following situations: 1) all response agents are located in the subtree of the initiator agent or 2) the ancestor between the initiator agent and all response agents is the lowest.*

According to the information exchange criterion in [25], our presented resource negotiation method can also have higher probability to reduce the communication cost compared to other random negotiation processes. Thus, we can obtain the good performance for resource negotiation, which is also tested by our simulation tests in Section 5.

Example 2 (continue). In Fig. 4, the set of resources owned by agent a_{21} is $\{r_1, r_1, r_2, r_3, r_3\}$. Now, a task t is allocated to a_{21}, and the resources requested by task t is $\{r_1, r_1, r_2, r_2, r_3, r_3, r_3, r_3\}$; obviously, a_{21} lacks the resources $\{r_2, r_3, r_3\}$. The negotiation process is shown in Fig. 5.

2.2.2 Arbitrary Structure

There are always two kinds of arbitrary social structures in the complex systems, one is the directed structure where agent interaction relation is unilateral, and the other is the undirected structure where agent interaction relation is bidirectional.

To minimize the negotiation communication time, we can make an agent negotiate with other agents for the requested resources based on the Breadth-First Traversal method in the

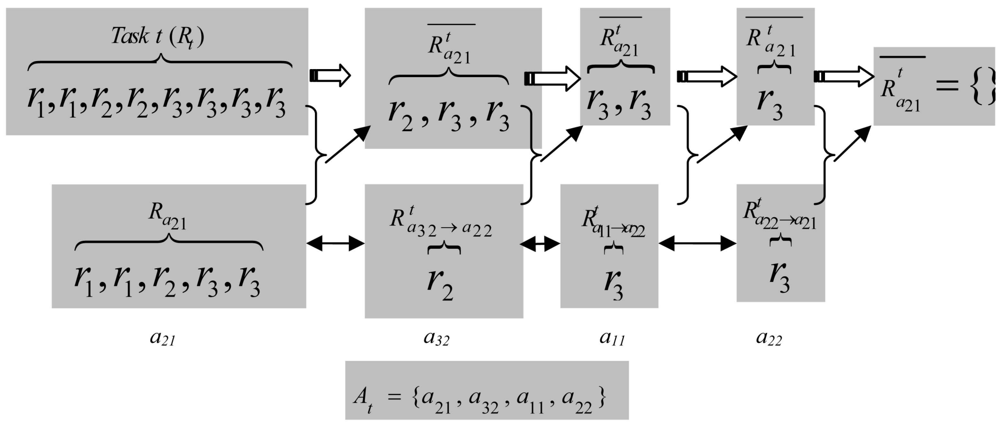
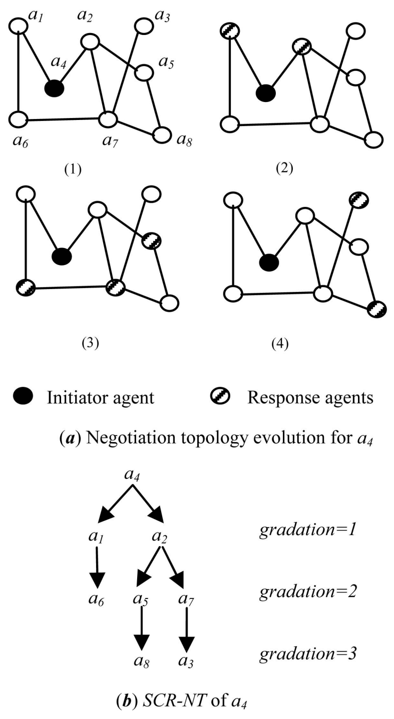
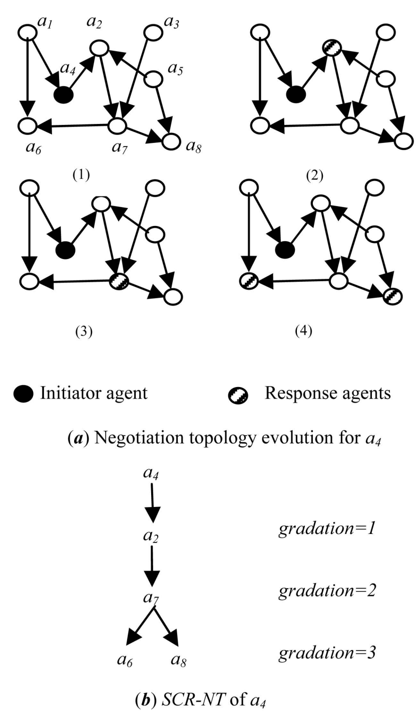


Fig. 5. Socially contextual resource negotiation process of the example in Fig. 4.



(a)

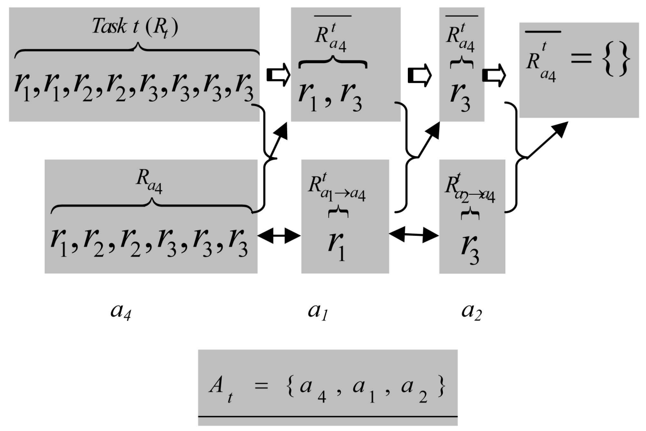


(b)

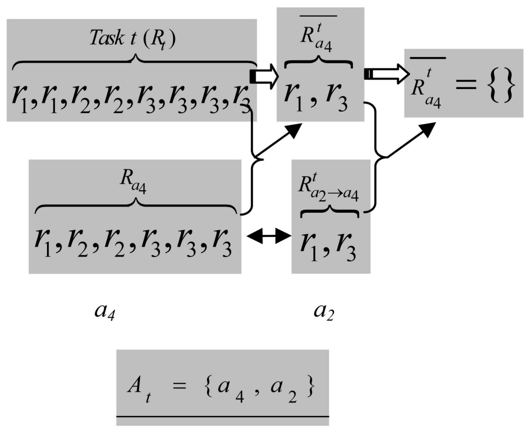
	r_1	r_2	r_3
a_1	2	2	0
a_2	2	1	2
a_3	1	2	1
a_4	1	2	3
a_5	2	1	0
a_6	2	0	0
a_7	1	2	1
a_8	1	2	0

(c)

Fig. 6. An example for constructing the *SCR-NT* in arbitrary social structure. (a) Undirected arbitrary social structure. (b) Directed arbitrary social structure. (c) The number of resources owned by the agents.



(a)



(b)

Fig. 7. Socially contextual resource negotiation process of the example in Fig. 6.

graph. The algorithm for the negotiation process within arbitrary social structure is the same as Algorithm 1.

Theorem 2. *If all resources for implementing task t can be satisfied by using our algorithm, the total communication cost between the initiator agent and the socially response agents in arbitrary social structure is the minimum.*

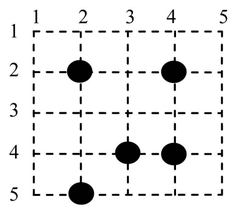
Proof. The proof is similar to that of Theorem 1, so here we skip it to save paper space. □

Example 3. Fig. 6 is an example to demonstrate the *SCR-NTs* of undirected and directed arbitrary structures. Fig. 7 is an example of the negotiation process of a_4 to implement task t. For the reason of saving space, we do not express the negotiation process in detail.

3 Task Allocation to Meet the Contextual Resource Distribution

3.1 Contextual Resource Enrichment Factor

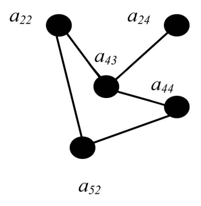
An agent will be more resource predominant if its contextual agents have more resources. Let there be an agent a_i, the set of agents within its physical context is PC_i and the set of agents within its social context is SC_i. Obviously, every agent within PC_i or SC_i will contribute differently to the resource predominance of a_i; the contribution of an agent within PC_i or SC_i to agent a_i is determined by the physical or social distance between such agent and a_i. Now, we have the concepts of physically contextual enrichment factor and socially contextual enrichment factor.



$$\begin{array}{c|ccccc} & a_{22} & a_{24} & a_{43} & a_{44} & a_{52} \\ \hline a_{22} & 1 & 2 & 3 & 4 & 3 \\ a_{24} & 2 & 1 & 3 & 2 & 5 \\ a_{43} & 3 & 3 & 1 & 1 & 2 \\ a_{44} & 4 & 2 & 1 & 1 & 3 \\ a_{52} & 3 & 5 & 2 & 3 & 1 \end{array}$$

Physical environment Physical distance matrix

(a)



$$\begin{array}{c|ccccc} & a_{22} & a_{24} & a_{43} & a_{44} & a_{52} \\ \hline a_{22} & 1 & 2 & 1 & 2 & 1 \\ a_{24} & 2 & 1 & 1 & 2 & 3 \\ a_{43} & 1 & 1 & 1 & 1 & 2 \\ a_{44} & 2 & 2 & 1 & 1 & 1 \\ a_{52} & 1 & 3 & 2 & 1 & 1 \end{array}$$

Social structure Social distance matrix

(b)

Agent	Number of self-owned resources			Physical enrichment factor			Social enrichment factor			Comprehensive enrichment factor		
	r_1	r_2	r_3	r_1	r_2	r_3	r_1	r_2	r_3	r_1	r_2	r_3
a_{22}	2	3	0	2.034	1.517	1.862	2.25	1.25	2.75	2.142	1.384	2.306
a_{24}	1	0	2	1.934	0.803	2.158	2.15	0.85	2.8	2.042	0.827	2.479
a_{43}	3	1	5	2.526	0.789	3.053	2.222	1	2.444	2.374	0.895	2.749
a_{44}	3	0	2	2.486	0.676	3.027	2.375	0.875	3	2.431	0.776	3.014
a_{52}	2	1	4	2.268	1.056	3.197	2.304	1.174	2.391	2.286	1.115	2.794

(c)

Fig. 8. An example for computing the contextual resource enrichment factors of agents, where PCs and SCs of every agent are both the whole agent set A: $\{a_{22}, a_{24}, a_{43}, a_{44}, a_{52}\}$. For the reason of simplicity, we can let $\lambda_s = \lambda_p = 0.5$. (a) Physical context. (b) Social context. (c) The contextual resource enrichment factors

Definition 5. *The* physically contextual enrichment factor *of agent a_i for resource r_k is*

$$\Phi_i(k) = \sum_{a_j \in PC_i} \left(n_j(k) \cdot \frac{1/d^p_{ij}}{\sum_{a_j \in PC_i} 1/d^p_{ij}} \right), \tag{4}$$

where $n_j(k)$ is the number of resource r_k owned by agent a_j, d^p_{ij} is the physical distance between a_i and a_j; $a_i \in PC_i$.

Definition 6. *The* socially contextual enrichment factor *of agent a_i for resource r_k is*

$$\Psi_i(k) = \sum_{a_j \in SC_i} \left(n_j(k) \cdot \frac{1/d^s_{ij}}{\sum_{a_j \in SC_i} 1/d^s_{ij}} \right), \tag{5}$$

where d^s_{ij} is the social distance between agents a_i and a_j. We can see that d^s_{ij} denotes the contribution weight of agent a_j to the enrichment factor of a_i, and d^s_{ij} appears both in the numerator and denominator in (5); thus, we can abstract d^s_{ij} into a natural number. If two agents are adjacent with each other in the interaction structure, they can make resource negotiation directly and their social distance can be set to 1; therefore, the social distance between two agents can be denoted by the shortest path length in the interaction structure.

Definition 7. *The* comprehensive contextual enrichment factor *of agent a_i for resource r_k is determined corporately by the physically and socially contextual enrichment factors:*

$$\begin{aligned} \xi_i(k) &= \lambda_p \Phi_i(k) + \lambda_s \Psi_i(k) \\ &= \lambda_p \sum_{a_j \in PC_i} \left(n_j(k) \cdot \frac{1/d^p_{ij}}{\sum_{a_j \in PC_i} 1/d^p_{ij}} \right) \\ &\quad + \lambda_s \sum_{a_j \in SC_i} \left(n_j(k) \cdot \frac{1/d^s_{ij}}{\sum_{a_j \in SC_i} 1/d^s_{ij}} \right), \end{aligned} \tag{6}$$

where λ_P and λ_S are used to determine the relative importance of the two kinds of contexts in the contextual enrichment factor of an agent, $\lambda_P + \lambda_S = 1$.

For an agent, the higher its contextual resource enrichment factor to one kind of resource is, the higher is its chance to obtain enough of such kind of resource in the system.

Example 4. Fig. 8 is an example of contextual resource enrichment factors of agents.

3.2 Contextual Resource-Based Task Allocation

The agents are complex with multiple resources related to the multiple tasks they implement in the complex systems. An optimal allocation in a complex system is to maximize the dependability that the tasks can obtain enough resources to be implemented.

For example, in Fig. 8, let there be a task that needs resource r_1. If we implement the task allocation based on the physically contextual resource distribution, we should allocate the task to agent a_{43}; if we implement the task

TABLE 1
An Example of Task Allocation According to the Comprehensive Enrichment Factor

Task	Required resources and number	*FRFS*			*MIFS*			*AAS*		
		PA	*CA*		*PA*	*CA*		*PA*	*CA*	
			PN	*SN*		*PN*	*SN*		*PN*	*SN*
t_1	$3r_1$, $1\mathbf{r_3}^*$, $4r_2$	a_{44} $(3r_1,1r_3)$	$a_{43}(1r_2)$ $a_{52}(1r_2)$ $a_{22}(2r_2)$	$a_{43}(1r_2)$ $a_{52}(1r_2)$ $a_{22}(2r_2)$	a_{44} $(3r_1,1r_3)$	$a_{43}(1r_2)$ $a_{52}(1r_2)$ $a_{22}(2r_2)$	$a_{43}(1r_2)$ $a_{52}(1r_2)$ $a_{22}(2r_2)$	a_{44} $(3r_1,1r_3)$	$a_{43}(1r_2)$ $a_{52}(1r_2)$ $a_{22}(2r_2)$	$a_{43}(1r_2)$ $a_{52}(1r_2)$ $a_{22}(2r_2)$
t_2	$2r_3$, $3\mathbf{r_1}$, $3r_2^*$	a_{44} $(2r_3,3r_1)$	$a_{43}(1r_2)$ $a_{52}(1r_2)$ $a_{22}(1r_2)$	$a_{43}(1r_2)$ $a_{52}(1r_2)$ $a_{22}(1r_2)$	a_{22} $(2r_1,3r_2)$	a_{24} $(2r_3,1r_1)$	a_{43} $(2r_3,1r_1)$	a_{44} $(2r_3,3r_1)$	$a_{43}(1r_2)$ $a_{52}(1r_2)$ $a_{22}(1r_2)$	$a_{43}(1r_2)$ $a_{52}(1r_2)$ $a_{22}(1r_2)$
t_3	$2r_2^*$, $3\mathbf{r_1}$, $1r_3$	a_{22} $(2r_2,2r_1)$	a_{24} $(1r_1,1r_3)$	a_{43} $(1r_1,1r_3)$	a_{22} $(2r_2,2r_1)$	a_{24} $(1r_1,1r_3)$	a_{43} $(1r_1,1r_3)$	a_{44} $(3r_1,1r_3)$	$a_{43}(1r_2)$ $a_{52}(1r_2)$	$a_{43}(1r_2)$ $a_{52}(1r_2)$

* *The most important resource. PA: principal agent; CA: cooperated agent; PN: physical negotiation; and SN: social negotiation.*

allocation based on the socially contextual resource distribution, we should allocate the task to agent a_{44}; if we implement the task allocation based on the comprehensive contextual resource distribution, we should allocate the task to agent a_{44}.

From the above example, we can see that it is easy to make definite effective allocation if the task only requires one kind of resources. However, if a task needs many kinds of resources, then how can we implement the allocation? For example, in Fig. 8, we let a task need resource r_1 and r_2. If we implement the task allocation based on comprehensive contextual resource distribution, a_{44} has the highest enrichment factor for r_1 and a_{22} has the highest enrichment factor for r_2. Then, which one of a_{44} or a_{22} should be allocated with the task? To solve this problem, we can design the allocation method for the task that requires more than one kind of resources as follows:

When a task requires more than one resource, we should assign an agent to act as the *principal one* (i.e., the initiator agent within the negotiation), which will initially negotiate with other agents for the resources required by the task; then, the set of all agents that provide resources for the task is the allocated agents. To determine the principal agent, we set some criterions as follows:

1. *First Required Resource-First Satisfy (FRFS).* When a task requires many resources, it may call each resource sequentially. Now, we can design the criterion of *FRFS*, i.e., the agent that has the highest enrichment factor for the first called resource should be allocated as the principal one of that task. Let there be a task t, the set of resources called by task t orderly is $\{n_1r_1, n_2r_2, \ldots, n_nr_n\}$; n_i denotes the required number of resource r_i. So, we will allocate t to the agent a_j, which has the highest enrichment factor for r_1, i.e., the highest $\Phi_j(1)$, $\Psi_j(1)$, or $\xi_j(1)$. For example, in Fig. 8, if the called resource sequence of a task is $\{r_2, r_1, r_2, r_3\}$ and a_{22} has the highest contextual resource enrichment factor for resource r_2, a_{22} will be assigned as the principal one for such task.
2. *Most Important Resource-First Satisfy (MIFS).* When a task requires many resources, there may be one resource that is the most important for implementing that task. Now, let there be a task t, the set of resources called by task t orderly is $\{n_1r_1, n_2r_2, \ldots, n_nr_n\}$; if r_i is the most important resource for implementing t, we will allocate t to agent a_j, which has the highest enrichment factor for resource r_i, i.e., the highest $\Phi_j(i)$, $\Psi_j(i)$, or $\xi_j(i)$. For example, in Fig. 8, if the called resource sequence of a task t is $\{r_2, r_1, r_2, r_3\}$ and r_1 is the most important for the implementation of t, a_{44} will be assigned as the principle one for t since a_{44} has the highest contextual enrichment factor for resource r_1.
3. *All Resources-Averagely Satisfy (AAS).* When a task t requires resources R_t, we can allocate the task to the agent, which has the highest average contextual enrichment factor for all resources in R_t. Now, let $R_t = \{n_1r_1, n_2r_2, \ldots, n_nr_n\}$, we will allocate task t to the following agent (according to the comprehensive contextual resource enrichment factor $\xi_j(i)$):

$$a_* = \arg\max_{a_j \in A}\left(\frac{1}{|R_t|}\sum_{r_i \in R_t} \xi_j(i)\right). \tag{7}$$

Therefore, our **task allocation process** is explained as follows: 1) Assign task t to a principal agent according to the above three criterions, now it is assumed that the principal agent is a_*; 2) let the set of resources required by task t be R_t, the resources owned by agent a_* be R_{a_*}, and $\overline{R^t_{a_*}} = R^t - R_{a_*}$, now agent a_* will behave as the initiator agent and negotiate with other agents within its physical context or social context according to the methods presented in Section 2; and 3) let the set of agents that will provide the requested resources to a_* be A', thus task t will be finally implemented by the allocated agent set: $A_t = \{a_*\} \cup A'$.

Example 4 (continue). We use the example in Fig. 8 to demonstrate the task allocation, as shown in Table 1. For example, task t_2 needs three kinds of resources: $2r_3$, $3r_1$, and $3r_2$; if we use *MIFS* to determine the principal agent, a_{22} will be selected and provide $2r_1$ and $3r_2$ for the task; so now a_{22} will negotiate with other contextual agents for $2r_3$ and $1r_1$; if a_{22} negotiates with other agents within its physical context, the agent set $\{a_{24}\}$ will be allocated; if a_{22} negotiates with other agents within its social context, the agent set $\{a_{43}\}$ will be allocated.

4 Context-Associated Load Balancing for Multiple Tasks

According to the model in Section 3, an agent may have more tasks to queue if it has higher contextual resource enrichment factor. Therefore, we should deal with the *load balancing* while there are multiple tasks within the complex system.

Generally, the performance of load balancing is mainly determined by the waiting time of the task. According to Liu et al. [10], the effect of measures on load balancing is reflected by the number and size of task teams on agents; and we simply consider that the waiting time is only related to the length of the task team. Let the allocated agent set of task t be A_t and the set of resources required by t be R_t. We can denote the team of tasks that queue for the resource r_k of agent a_i as Q_{ik}, and the size of Q_{ik} as s_{ik}. If task t calls its requested resources concurrently, the waiting time of task t will be determined by the maximum size of the queues for its required resources; if task t calls its requested resources serially, the waiting time of task t will be determined by the total sizes of the queues for its required resources. Therefore, we can define the waiting time of a task as

$$w_t = \max_{\forall r_i \in R_t, \forall a_k \in A_t} (s_{ik}), \quad \text{or} \quad w_t = \sum_{\forall r_i \in R_t, \forall a_k \in A_t} (s_{ik}). \tag{8}$$

Now, if there are multiple tasks that arrive at the complex system, the set of those multiple tasks is T, thus we can define the global load balancing performance of the system as

$$\widetilde{w} = \frac{1}{|T|} \sum_{t \in T} w_t. \tag{9}$$

Therefore, our goal of load balancing is to reduce the average size of the queues for all resources and all agents. In the previous research work on load balancing [10], [11], [28], an agent's probability to be allocated new tasks will be reduced if it has already undertaken too many tasks; thus, the number of undertaken tasks of an agent will only influence the probability of itself to be allocated new tasks in the task allocation.

However, an agent will negotiate with its contextual agents for resources; thus, if an agent has already been allocated with too many tasks, the probabilities of its contextual agents to accept new tasks should also be reduced. Thus, we will develop the load balancing method also based on the contextual resource distribution. If an agent, a, is allocated with some tasks, it is not only a itself but also the agents in the context of a will be reduced the possibilities to accept new tasks in the future. Let s_{ik} be the size of team where tasks queue for the resource r_k of agent a_i, the contextual enrichment factors of agents within the physical and social contexts of a_i will be changed as

$$\forall a_j \in PC_i, \Phi_j^*(k) = \Phi_j(k) - \sigma\left(\frac{s_{ik}}{d_{ij}^p}\right), \tag{10}$$

$$\forall a_j \in SC_i, \Psi_j^*(k) = \Psi_j(k) - \sigma\left(\frac{s_{ik}}{d_{ij}^s}\right), \tag{11}$$

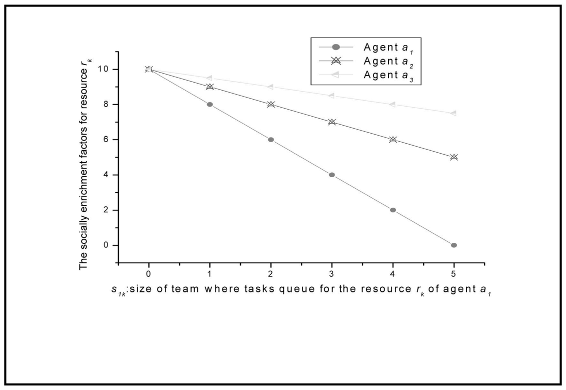


Fig. 9. Numerical simulation for the contextual resource enrichment factor modification.

where σ is a monotonously increasing function; d_{ij}^p is the physical distance between agents a_i and a_j, $a_i \in PC_i$; d_{ij}^s is the social distance between agents a_i and a_j, $a_i \in SC_i$.

Example 4 (numerical simulations). Let there be three agents in a social context, $\{a_1, a_2, a_3\}$, the social distance from a_1 to a_2 is 1, the social distance from a_1 to a_3 is 2, for simplicity, we can let the social distance from a_1 to itself be 0.5; Let the size of team where tasks queue for the resource r_k of agent a_1, s_{1k}, be 0 initially, and the socially contextual enrichment factors of the three agents to r_k be all 10 initially. Now, the value of s_{1k} increases with the step of 1; from Fig. 9, we can see that the enrichment factors of three agents will decrease accordingly; and the nearer to a_1, the more rapidly the agent's enrichment factor decreases.

5 Validations and Analyses

To validate the correctness and effect of our task allocation and load balancing model based on contextual resource negotiation, we make a series of case simulations. We can define the task in the simulation test as follows:

It is assumed that each resource of an agent can only be obtained by one task at the same time. Let the task is allocated to a series of agents, and the task creates a token that will be sent from the first agent to the next and so on along the agent series until it arrives at the last agent and obtains the required resources. Thus, the execution of task is given as follows: 1) First, the token is dead, 2) when the token is newly received by an allocated agent, then it will be dead; only when it can obtain the enough required resources from its now inhabited agent, the token can be activated and sent to the next allocated agent, and 3) such process will continue until the token visits all allocated agents and is active at last.

For example, let task t_1 be allocated to "$a_{44}(3r_1, 1r_3)$, $a_{22}(3r_2)$, $a_{43}(1r_2)$," so the execution of t_1 is given as follows: 1) At first, a dead token is created in a_{44}, when t_1 obtains "$3r_1$, $1r_3$" from a_{44}, the token is activated and sent to a_{22}, then resources "$3r_1$, $1r_3$" are released; 2) now, a_{22} receives the token and makes it a dead one, when t_1 obtains "$3r_2$" from a_{22}, the token is activated and sent to a_{43}, then resources "$3r_2$" are released; 3) now, a_{43} receives the token and makes it a dead one; when t_1 obtains "$1r_2$" from a_{43}, the token is activated and now the task is finished, and resources "$1r_2$" are released. Therefore, let g_{ai} denote the time of "getting required resources" from agent a_i, $c_{ai \to aj}$

TABLE 2
The Task Allocation Processes in the *SRM* Model

FRFS (First required resource-firstly satisfy)	*MIFS (Most important resource-firstly satisfy)*	*AAS (All resources-averagely satisfy)*

1). A_t={};
2).**While** $R_t \neq \{\}$:
  2.1). $r*$ *be the first required resource in* R_t;
  2.2). $a*=\text{argmax}_{aj\in A}(n_j(*))$;
  2.3). $A_t = A_t \cup \{a*\}$;
  2.4). $R_t = R_t - R*$;
3). **Output** (A_t).
	1). A_t={};
	2).**While** $R_t \neq \{\}$:
	  2.1). $r*$ *be the most important resource in* R_t;
	  2.2). $a*=\text{argmax}_{aj\in A}(n_j(*))$;
	  2.3). $A_t = A_t \cup \{a*\}$;
	  2.4). $R_t = R_t - R*$;
	3). **Output** (A_t).
		1). A_t={};
		2).**While** $R_t \neq \{\}$:
		  2.1). $a_* = \arg\max_{a_j \in A}(\frac{1}{\mid R_t \mid}\sum_{r_i \in R_t} n_j(i))$;
		  2.2). $A_t = A_t \cup \{a*\}$;
		  2.3). $R_t = R_t - R*$;
		3). **Output** (A_t).

denote the communication time from a_i to a_j, then the execution time of t_1 is

$$E_{t1} = g_{a44} + c_{a44\to a22} + g_{a22} + c_{a22\to a43} + g_{a43}. \qquad (12)$$

From above, we can see that the execution time of a task is determined by the resource access time (i.e., the waiting time of the task described in Section 4) and the communication time among allocated agents. Therefore, the task team on an agent is longer, the time of "getting resource" is longer, and that leads the execution time of task to become longer; on the other hand, the distance among allocated agents is longer, the time of "communication time" will be longer, and that also leads the execution time of task to become longer.

5.1 Comparison between Contextual Resource-Based Task Allocation and Self-Owned Resource-Based Task Allocation

In this section, we will validate the correctness of our task allocation model, so we will compare the performances of the following two models: 1) Contextual resource-based task allocation model, *CRM* and 2) Self-owned resource-based task allocation model, *SRM*. In the *SRM*, the task allocation is implemented according to the self-owned resources of agents themselves, which is always used in the previous related work. Now, we introduce the *SRM* model briefly.

When a task requires more than one resource, we can also use the three criterions described in Section 3.2. Now, let there be a task t, the set of resources called by task t is $R_t = \{n_1r_1, n_2r_2, \ldots, n_nr_n\}$, $n_j(i)$ denotes the number of r_i owned by agent a_j, R_i denotes the set of resources owned by agent a_i, then the task allocation processes of *SRM* are shown in Table 2.

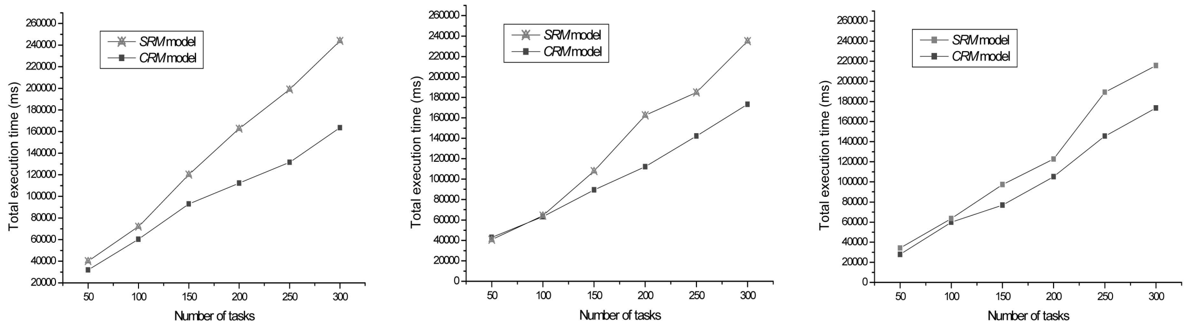
Then, we make a series of case simulations to test the two models. In the simulations, the agent number is 200, the physical distribution and social structure can be set randomly. We can use *SRM* and *CRM* models to make task allocation; the allocated agents will execute the tasks, then the total execution time of tasks is tested. The results are shown in Fig. 10.

From the results, we can obtain the following: 1) when *CRM* model is used, the allocated agents will always incline to be located within the near physical contexts or social contexts, so the communication time will be reduced; however, the allocated agents in *SRM* model will always be distributed through the system, so the communication time among agents will incline to be more than the one of *CRM* model. Therefore, the task execution time of *CRM* model is always less than the one of the *SRM* model and 2) while the number of tasks increases, the communication time will also increase; so the difference between the two models will increase accordingly.

Therefore, we can conclude that the *CRM* model can reduce the total communication time among allocated agents so as to reduce the total task execution time compared to *SRM*, especially while the number of tasks is large.

5.2 Comparison between Contextual Resource-Based Load Balancing and Self-Owned Resource-Based Load Balancing

Now, we will compare the contextual resource-based load balancing and the self-owned resource-based load balancing.



(a) (b) (c)

Fig. 10. Execution time comparison between *SRM* and *CRM* models. (a) FRFS. (b) MIFS. (c) AAS.

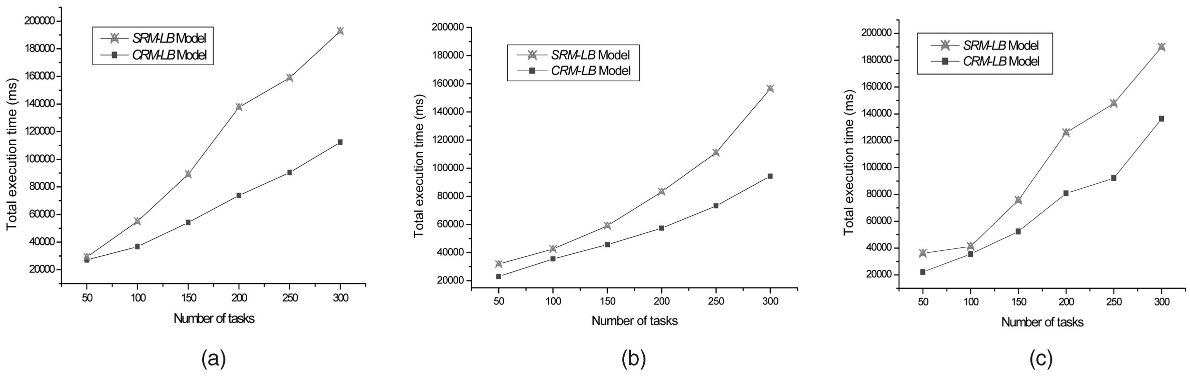


Fig. 11. Execution time comparison between *SRM-LB* and *CRM-LB* models. (a) FRFS. (b) MIFS. (c) AAS.

In Section 4, we have addressed the contextual resource-based task allocation with load balancing (*CRM-LB*), now we briefly introduce the self-owned resource-based task allocation with load balancing (*SRM-LB*).

Let s_{jk} be the size of teams where tasks queue for the resource r_k of agent a_j, the number of r_k owned by agent a_j is $n_j(k)$, so the self-owned resource enrichment factor of agent a_j on r_k, $\varphi_j(k)$, should be changed as (13) when we consider the load balancing:

$$\varphi_j(k) = n_j(k) - f(s_{jk}), \tag{13}$$

where f is a monotonously increasing function. Therefore, while an agent is allocated with more tasks, its probability to obtain new task in the future will be reduced. When a task t is allocated to some agents that are A_t, all agents in A_t should modify their self-owned resource enrichment factor according to (13).

We make some case simulations to test the two models. In the simulations, the agent number is 200, the physical distribution and social structure can be set randomly. We can use the *CRM-LB* model and *SRM-LB* model to make the task allocation; the allocated agents will execute the tasks, then the total execution time for the tasks is tested. The results are shown in Fig. 11.

From the results, we can obtain the following:

1. The total execution time in *CRM-LB* is less than the one in *SRM-LB*, since the *CRM-LB* model can reduce the total communication time among allocated agents so as to reduce the total task execution time that is better than *SRM-LB*, especially while the number of tasks is large.
2. The gap between the two models in Fig. 11 is bigger than the one in Fig. 10, so the load balancing in *CRM-LB* outperforms the one in *SRM-LB*. We conclude that *CRM-LB* is better suited for the characteristics of complex software systems, since *CRM-LB* adjusts the contextual resource enrichment factors of allocated agents themselves as well as their contextual agents, but *SRM-LB* only adjusts the self-owned resource enrichment factors of allocated agents themselves.

5.3 Comparison between CRM-LB and CRM

We now compare *CRM-LB* and the contextual resource-based task allocation *without* load balancing (*CRM*). In the simulations, the agent number is 200, the physical distribution and social structure can be set randomly. We can use the *CRM-LB* and *CRM* models to make the task allocation; the allocated agents will execute the task, then the total execution time for the tasks is tested. The results are shown in Fig. 12.

From the results, we can see that *CRM-LB* outperforms *CRM*; while the task number increases, the *CRM-LB* will outperform *CRM* more and more. Therefore, our load balancing model can make an effective load balancing and reduce the task waiting time for resources while there are multiple tasks in the complex systems.

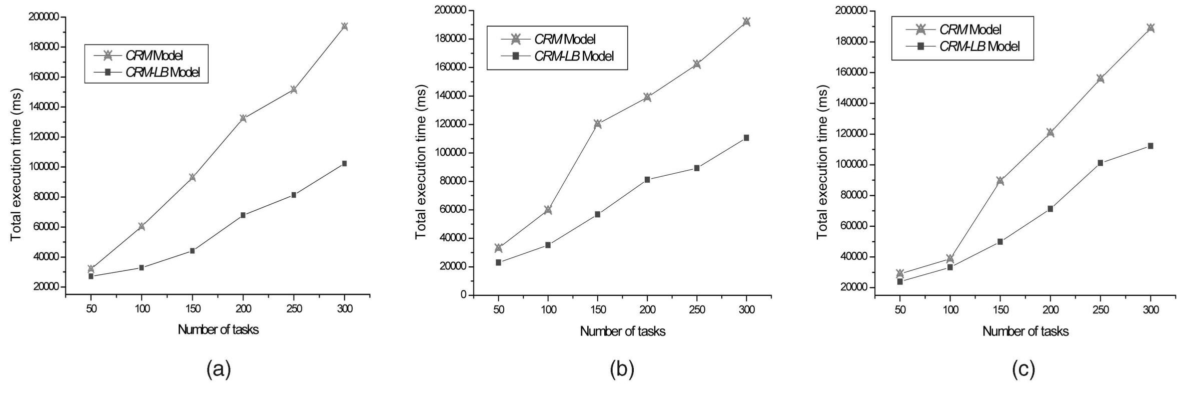


Fig. 12. Execution time comparison between *CRM* and *CRM-LB* models. (a) FRFS. (b) MIFS. (c) AAS.

6 Discussion

In our model, the agents are assumed to only differ essentially in their resources and they have the same computable functions; thus, the task allocation and load balancing are implemented only based on the contextual resource distribution. In some real situations, such assumption is reasonable since all agents are identical. For example, in a hotel-booking system, all booking agents may have the same functions but different hotel information; thus, the booking tasks are always assigned to the agents with rich hotel information.

However, in many circumstances, the above assumption does not match the peculiarities of real multiagent systems; in real systems, some agents may have different computable functions as well as different resources. Therefore, now we should make task allocation and load balancing by considering the resources together with computable functions.

First, if we only consider the computable functions, now the model based on computable function negotiation can be explained in brief as follows:

If an agent, a, does not have necessary computable function to implement a task by itself, whereas its contextual agent has such function, with our model, a may ***entrust*** the task to its contextual agent to implement; thus, the number of allocated tasks on an agent is directly proportional to not only its own computable functions but also the ones of its contexts. (*The model based on contextual function negotiation*.)

Therefore, with the model based on contextual function negotiation, we only need to modify the contextual resource enrichment factors in Section 3 to the contextual function enrichment factors; then, the task allocation and load balancing can be implemented according to the contextual function enrichment factors of all agents. Obviously, all algorithms in the new model are the same as the model based on contextual resource negotiation.

Moreover, if we want to make task allocation based on both resources and functions of agents, we can further extend our presented model by only further modifying the contextual resource enrichment factors. Now, we can present the concept of *contextual resource and function (RF) enrichment factor*, which makes trade-off between the resource difference and the function difference. If an agent has higher contextual RF enrichment factor, it may have higher probability to obtain tasks.

Therefore, despite our presented model assumes that all agents have the same computable functions and differ in resources only, the model can easily be extended into other real situations where agents have different computable functions as well as resources.

7 Conclusion

This paper has presented a novel model for task allocation and load balancing based on contextual resource negotiation. The presented model can be well suited for the characteristics of complex software systems, thus it outperforms the previous methods based on the self-owned resource distribution of agents. With the model, the communication costs between agents to execute the allocated task are reduced, so the total execution time of tasks can also be reduced accordingly. Through the simulations, we can see that the model can work well especially while the task number is large. Therefore, the idea presented in this paper can be used to develop real large-scale complex systems. Moreover, we recognize that how to make trade-off between physical and social negotiations should be explored in our future work since we know that a good trade-off between the two negotiations can bring benefit for us [16].

Acknowledgments

This research was supported by the National Natural Science Foundation of China (No. 60803060), the National High Technology Research and Development Program of China (863 Program) (No. 2009AA01Z118), the Excellent Young Teachers Program of Southeast University (No. 4050181013), the Specialized Research Fund for the Doctoral Program of Higher Education (No. 200802861077), and the Scientific Research Foundation for the Returned Overseas Chinese Scholars, State Education Ministry of China. The author would like to thank the associate editor, Dr. Rudolf Eigenmann, for his timely handling of the submission and the three anonymous reviewers for their constructive comments and suggestions.

References

[1] H.A. Simon, *The Sciences of the Artificial.* MIT Press, 1996.

[2] J. Liu, X. Jin, and K.C. Tsui, *Autonomy Oriented Computing (AOC): From Problem Solving to Complex Systems Modeling.* Kluwer Academic/Springer, 2005.

[3] J. Liu, X. Jin, and K.C. Tsui, "Autonomy-Oriented Computing (AOC): Formulating Computational Systems with Autonomous Components," *IEEE Trans. Systems, Man, and Cybernetics—Part A: Systems and Humans,* vol. 35, no. 6, pp. 879-902, Nov. 2005.

[4] N.R. Jennings, "An Agent-Based Approach for Building Complex Software Systems," *Comm. ACM,* vol. 44, no. 4, pp. 35-41, 2001.

[5] N.R. Jennings, "Agent-Oriented Software Engineering," *Lecture Notes on Computer Science,* vol. 1611, pp. 4-10, 2004.

[6] O. Shehory and S. Kraus, "Methods for Task Allocation via Agent Coalition Formation," *Artificial Intelligence,* vol. 101, nos. 1-2, pp. 165-200, 1998.

[7] Y.-C. Jiang and J.C. Jiang, "A Multi-Agent Coordination Model for the Variation of Underlying Network Topology," *Expert Systems with Applications,* vol. 29, no. 2, pp. 372-382, 2005.

[8] S. Kraus and T. Plotkin, "Algorithms of Distributed Task Allocation for Cooperative Agents," *Theoretical Computer Science,* vol. 242, nos. 1-2, pp. 1-27, 2000.

[9] S. Kraus, "Negotiation and Cooperation in Multi-Agent Environment," *Artificial Intelligence,* vol. 94, no. 1, pp. 79-97, 1997.

[10] J. Liu, X. Jin, and Y. Wang, "Agent-Based Load Balancing on Homogeneous Minigrids: Macroscopic Modeling and Characterization," *IEEE Trans. Parallel and Distributed Systems,* vol. 16, no. 7, pp. 586-598, July 2005.

[11] K.-P. Chow and Y.-K. Kwok, "On Load Balancing for Distributed Multiagent Computing," *IEEE Trans. Parallel and Distributed Systems,* vol. 13, no. 8, pp. 787-801, Aug. 2002.

[12] X. Zhang, V. Lesser, and T. Wagner, "Integrative Negotiation among Agents Situated in Organizations," *IEEE Trans. Systems, Man, and Cybernetics—Part C: Applications and Rev.,* vol. 36, no. 11, pp. 19-30, 2006.

[13] H. Haken, *Information and Self-Organization: A Macroscopic Approach to Complex Systems,* second ed. Springer, Jan. 2000.

[14] K.J. Dooley, "A Complex Adaptive Systems Model of Organization Change," *Nonlinear Dynamics, Psychology, and Life Sciences,* vol. 1, no. 1, pp. 69-97, 1997.

[15] S. Kalenka, "Modeling Social Interaction Attitudes in Multi-Agent Systems," PhD thesis, Dept. of Electronic Eng., Queen Mary, Univ. of London, 2001.

[16] Y. Jiang, J. Jiang, and T. Ishida, "Compatibility between the Local and Social Performances of Multi-Agent Societies," *Expert Systems with Applications,* vol. 36, no. 3, part 1, pp. 4443-4450, 2009.

[17] Y. Jiang and T. Ishida, "A Model for Collective Strategy Diffusion in Agent Social Law Diffusion," *Proc. 20th Int'l Joint Conf. Artificial Intelligence (IJCAI '07),* Jan. 2007.

[18] Y. Jiang and T. Ishida, "Local Interaction and Non-Local Coordination in Agent Social Law Diffusion," *Expert Systems with Applications,* vol. 34, no. 1, pp. 87-95, 2008.

[19] A. Schaerf, Y. Shoham, and M. Tennenholtz, "Adaptive Load Balancing: A Study in Multi-Agent Learning," *J. Artificial Intelligence Research,* vol. 2, pp. 475-500, 1995.

[20] J. Bigham and L. Du, "Cooperative Negotiation in a Multi-Agent System for Real-Time Load Balancing of a Mobile Cellular Network," *Proc. Second Int'l Joint Conf. Autonomous Agents and Multiagent Systems (AAMAS '03),* pp. 568-575, 2003.

[21] J. Guo and L.N. Bhuyan, "Load Balancing in a Cluster-Based Web Server for Multimedia Applications," *IEEE Trans. Parallel and Distributed Systems,* vol. 17, no. 11, pp. 1321-1334, Nov. 2006.

[22] S. Dhakal, M.M. Hayat, J.E. Pezoa, C. Yang, and D.A. Bader, "Dynamic Load Balancing in Distributed Systems in the Presence of Delays: A Regeneration-Theory Approach," *IEEE Trans. Parallel and Distributed Systems,* vol. 18, no. 4, pp. 485-497, Apr. 2007.

[23] Y. Jiang, "Extracting Social Laws from Unilateral Binary Constraint Relation Topologies in Multiagent Systems," *Expert Systems with Applications,* vol. 34, no. 3, pp. 2004-2012, 2008.

[24] F. Wu, B.A. Huberman, L.A. Adamic, and J.R. Tyler, "Information Flow in Social Groups," *Physica A: Statistical and Theoretical Physics,* vol. 337, no. 1/2, pp. 327-335, June 2004.

[25] P.S. Dodds, D.J. Watts, and C.F. Sabel, "Information Exchange and the Robustness of Organizational Networks," *Proc. Nat'l Academy of Sciences of the US,* Oct. 2003, doi:10.1073/pnas.1534702100.

[26] A. Dey and G. Abowd, "Towards a Better Understanding of Context and Context-Awareness," *Proc. CHI Workshop What, Who, Where, When and How of Context-Awareness,* Apr. 2000.

[27] P. Faratin, C. Sierra, and N.R. Jennings, "Negotiation Decision Functions for Autonomous Agents," *J. Robotics and Autonomous Systems,* vol. 3-4, no. 24, pp. 159-182, 1998.

[28] Y. Le Quéré, M. Sevaux, C. Tahon, and D. Trentesaux, "Reactive Scheduling of Complex System Maintenance in a Cooperative Environment with Communication Times," *IEEE Trans. Systems, Man, and Cybernetics—Part C: Applications and Rev.,* vol. 33, no. 2, pp. 225-234, May 2003.

Yichuan Jiang received the PhD degree in computer science from Fudan University, Shanghai, China. He is currently a professor in the Lab for Knowledge Engineering and Social Computing, Research Center for Learning Science, Southeast University, Nanjing, China. From July to October 2005, he had been a postdoctoral researcher in the Department of Computer Science, Hong Kong Baptist University; and then he was a JSPS postdoctoral research fellow in the Department of Social Informatics, Kyoto University, Kyoto, Japan. His main research interests include multiagent systems, distributed computing, social intelligence, and applied artificial intelligence. He has published more than 40 scientific articles in refereed journals and conference proceedings. He is a member of IEEE and the IEEE Computer Society.

Jiuchuan Jiang received the bachelor's degree in communication engineering from Xiangtan University, Xiangtan, China, in 2004, and the master's degree in computer science from Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2009. He is currently a PhD student in the School of Information Science and Engineering, Southeast University, Nanjing, China. He had been an engineer in the Hunan Branch, China United Telecommunications Corporation, from 2004 to 2005. His main research interests include multiagent systems, distributed computing and complex systems.