ELSEVIER

Expert Systems with Applications 29 (2005) 372–382

Expert Systems with Applications

www.elsevier.com/locate/eswa

A multi-agent coordination model for the variation of underlying network topology

Yi-Chuan Jiang[a,*], J.C. Jiang[b]

[a]Department of Computing & Information Technology, Centre of Networking and Information Engineering, Fudan University, Room 409, Yifu Building, Shanghai 200433, China

[b]Hunan Branch, China United Telecommunications Corporation, Changsha 410001, China

Abstract

In now multi-agent systems, the underlying networks are always dynamic and the network topologies are always changed in the operation. Therefore, the coordination of agents shall be adjusted for the dynamic network topology. Aiming at the dynamics of underlying network topology, a novel adaptive multi-agents coordination model is explored in this paper. In the paper, a series of algorithms for multi-agent task and resource negotiation are provided. The provided algorithms consider the factors of network topology and agent distribution, and can implement effective task allocation and resource negotiation for current network topology. Therefore, the adaptation of agent coordination for dynamic underlying network topology can be achieved, which is also proved by the case studies and performance analyses in the paper.

Keywords: Multi-agent; Coordination; Network topology; Task allocation; Resource negotiation

1. Introduction

In the multi-agent system, the autonomous agents coordinate to execute the allocated task. An agent can profit from the actions of other agents, as well as benefit to other agents. Coordination computing cannot only make individual agent try its best but also combine agents to improve the ability of the overall system. Otherwise, coordination can save system resource and make the agent system more flexible.

Indeed, cooperation is often considered as one of the key concepts of agent communities (Buccaafurri, Rosaci, Sarne, & Palopoli, 2004). Automated intelligent agents inhabiting a shared environment must coordinate their activities. According to different system environments, such as the level of agents cooperation, agents regulation and protocols, the number and type of agents, and the communication cost, we can explore different coordination strategies in multi-agent systems (Kraus, 1997). But, whatever coordination strategy we explored, to realize effective agent cooperation, we should resolve the following two key problems: *task allocation* and *resource negotiation*. Only after solving them well, a real effective coordination strategy may be achieved.

When a multi-agent system wants to execute a task, the first step is to allocate the task to some agents, which is called *task allocation* (Billionnet, Costa, & Sutter, 1992; Tanaev, Sotskov, & Strusevich, 1994). The main problem in the task assignment is that no agent is versatile, and each agent has different capabilities and can only fulfil a subset of the tasks. Therefore, we should make an effective task allocation, i.e. get effective tasks-agents mapping strategy. The main goal of the task allocation is to maximize the overall performance of the system and to fulfil the tasks as soon as possible (Kraus & Plotkin, 2000).

There are some existing researches about the agent task allocation, which can be divided into central controlled fashion (e.g. Chang, Phiphobmongkol, & Day, 1993; Georgeff, 1983) and the distributed ones (e.g. Kai-Hsiung Chang & Day, 1990; Sandholm, 1993; Smith, 1980).

The central controlled fashion is simple to understand and implement, which mainly adopts a central managing agent to implement the task allocation, and the managing agent is assumed to have information about the constraints and capabilities of all other agents and knowledge about the domain environment (Chang et al., 1993).

* Corresponding author. Tel.: +86 2165643235; fax: +86 2165647894.
E-mail address: jiangyichuan@yahoo.com.cn (Y.-C. Jiang).

0957-4174/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2005.04.015

Therefore, the managing agent has to contain large storage of information, and may become the performance bottleneck.

In distributed task allocation fashion, agent can enact an auction among the agents and can match agents with tasks suing a cooperative bidding approach without any central authority (Palmer, Kirschenbaum, Murton, & Zajac, 2003; Shehory & Kraus, 1998). In distributed mechanism, there is no central authority that distributes the task among agents, and the agents shall reach an efficient task allocation by themselves, seeking a solution for to fulfil the constraints. A well-known example of the distributed task allocation is the Contract Net Protocol (e.g. Sandholm, 1993; Smith, 1980). Obviously, the distributed task allocation fashion does not produce a 'performance bottleneck' in the system. However, it is difficult to impose effective control on the task allocation process. Otherwise, the dynamics of network topology is difficult to manage in the distributed fashion.

Therefore, similar to Kraus and Plotkin (2000), we consider the problem of distributed dynamic task allocation by a collection agents with central controlled model, where only one distinguished agent (manager) knows about all of the agent distribution, the network topology, and the agent capabilities.

A coordination mechanism that works well in a reasonable static environment will often perform poorly in a dynamic and fast changing one (Excelente-Toledo & Jennings, 2004). Therefore, the adaptation of task allocation for dynamic environment should be addressed. Nowadays, the networks show themselves a new phenomenon, which is the *network topology dynamics*. In the operation of network, some nodes may enter or depart randomly, some links among nodes may be built or terminated randomly, and the distances among nodes may change randomly. Dynamic network topology modifications are essential for many reasons, e.g. to maintain a connection due to node mobility.

Aiming at the dynamics of the underlying network topology, the task allocation of multi-agent system should adapt itself according to the network topology. Therefore, in this paper, we introduce some concepts about the agent distribution and network topology, and present a series of algorithms for the task allocation for current network topology. The main difference between the other task allocation works and our problem is that the factors of underlying network topology and the agent distribution are considered in our work. Therefore, our solution can make the task allocation adapt to the underlying network topology and agent distribution.

After a task is allocated to a subset of agents, the agents require some resources to execute the task. However, the allocated agents may have not enough resources to finish the task but other agents may have redundant resources. Therefore, we should solve the agents' resource shortage conflicts and effectively accomplish tasks through agent resource negotiation. An agent can borrow resource from other agents by resource negotiation. However, a resource conflict can happen when more than one agent attempts to seek the same critical resource at the same time (Findler, 1995). In Findler (1995), N.V. Findler proposed a technique called 'hierarchical iterative conflict resolution' which resolves the conflicts in an iterative manner, based upon a hierarchy of task priorities. With that technique, agents with higher priority tasks may take resources belonging to agents with lower priority. However, the technique in Findler (1995) does not take the underlying network topology (e.g. geographically distance between agents) into account, therefore, it cannot perform well in the dynamic topology networks. In this paper, we consider the network topology and the geographically agent distribution, and present a series of algorithms for agent resource negotiation which solve the resource conflict according to not only the task priority but also the geographically agent distribution. Therefore, our algorithms can make the agent resource negotiation adapt for the network topology.

Agents should represent their knowledge about the environment in some forms, such as graph (Schuster, 2000), ontology (Gruber, 1991), etc. In this paper, we only consider the agent knowledge about the resource distribution in the network. By considering the resource ownership and location, we use the linked list to represent the knowledge of resource distribution, and present an autonomous agent resource negotiation mechanism. In our autonomous agent resource negotiation mechanism, agent can find the geographically nearest resource of the agent with lower priority task within its own knowledge about agent distribution, and agents can communicate their knowledge by the integration of linked list.

The rest of this paper is organized as follows. Section 2 presents the task allocation model and algorithms for dynamic topology networks. Section 3 addresses the resource negotiation in dynamic topology networks and presents the algorithms. Section 4 introduces the linked list to represent the resource knowledge of agent, and presents an autonomous resource negotiation mechanism. In Section 5, the case studies and performance analyses are described. Finally, Section 6 concludes the paper.

2. Task allocation for dynamic topology networks

2.1. Formal descriptions

Kraus and Plotkin (2000) gives the formal definitions of the environment for the distributed task allocation for cooperative agents. However, they do not deal with the agent distribution in the network and the network topology. To adapt for the dynamic topology network, now we extend the formal definitions in Kraus and Plotkin (2000), and present the formal definitions for the agent coordination in dynamic topology networks.

We consider a multi-agent system consisting of a set of agents A which are distributed on a network topology N;

the agents can perform different tasks, the set of all tasks is T; the execution of each task requires different capabilities of agents, each agent has some capabilities, the set of all agent capabilities are denoted as C.

Therefore, in the agent system environments with dynamic underlying topology network, we can assume that there are four kinds of binary relations:

(1) $\rho_1 \subset A \times C$ such that for any $a_i \in A$, $c_j \in C$, $a_i \rho_1 c_j$ holds iff agent a_i has the capability c_j;
(2) $\rho_2 \subset A \times N$ such that for any $a_i \in A$, $n_j \in N$, $a_i \rho_2 n_j$ holds iff agent a_i locates on the node n_j;
(3) $\rho_3 \subset T \times C$ such that for any $t_i \in T$, $c_j \in C$, $t_i \rho_3 c_j$ holds iff task t_i requires the capability c_j;
(4) $\rho_4 \subset T \times A$ such that for any $t_i \in T$, $a_j \in A$, $t_i \rho_4 a_j$ holds iff task t_i is allocated to the agent a_j;

In multi-agent systems, each agent has different capability, and each agent can locate on different host. To execute a task, some agent capabilities shall be satisfied, and the task shall be allocated to some agents.

Given $a_i, a_j \in A$, we denote by $C_{a_i}, C_{a_j} \subseteq C$ the set of capabilities that agent a_i and a_j has respectively. Two agents may have the same capability, thus it may be the case that for some $a_i, a_j \in A$, $C_{a_i} \cap C_{a_j} \neq \phi$.

Now we assume that there is a task $t_i \in T$ that arrives at the agent system, and t_i needs a set of capabilities $C_{t_i} \subseteq C$, the cardinality of C_{t_i} is n. For each $c_j \in C_{t_i} \subseteq C$, there may be several agents that have it. We denote by $A_{t_i}^{c_j} \subseteq A$ the set of agents that have the same capability c_j (c_j is required by task t_i). Obviously, it may be the case that for some $j_i \neq j_2$, $A_{t_i}^{c_{j_1}} \cap A_{t_i}^{c_{j_2}} \neq \phi$.

In the agent system, the agents are distributed on the network, therefore, the communication costs among different agent sets are different. If more than one agent have the same capability required by the task, we shall select an agent among them such that the communication cost between it and other agents (executing the task) is the minimum.

Therefore, let t_i be the task that arrives at the agent system, $C_{t_i} = \{c_1, c_2, \ldots, c_n\}$ be the capabilities set required by the task t_i, $A_{t_i}^{c_i}$ be the agents set that have the same capability c_i (which is required by t_i), the problem of *agent task allocation for dynamic topology* can be described as follows.

Finding a set of agents $A_r = \{a_r^1, a_r^2, \ldots, a_r^n\}$, where n is the cardinality of C_{t_i}, $a_r^1 \in A_{t_i}^{c_1}$, $a_r^2 \in A_{t_i}^{c_2}$, $\ldots a_r^n \in A_{t_i}^{c_n}$, so as to the communication among $a_r^1, a_r^2, \ldots, a_r^n$ is the minimum.

From above problem description, we can see that the agent distribution and the network topology (e.g. communication distance among agents) should be considered in the task-allocation.

For example, we can suppose a task t needs four capabilities, $C_t = \{C_1, C_2, C_3, C_4\}$. The set of agents that have the capability c_1 is: $A_t^{c_1} = \{a_1, a_2, a_3\}$; the set of agents that have the capability c_2 is: $A_t^{c_2} = \{a_1, a_4, a_6, a_7\}$; the set of agents that have the capability c_3 is: $A_t^{c_3} = \{a_5, a_6, a_9\}$;

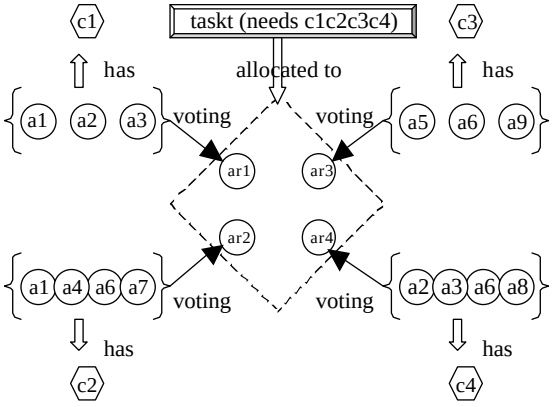

Fig. 1. An example of the formal descriptions.

the set of agents that have the capability c_4 is: $A_t^{c_4} = \{a_2, a_3, a_6, a_8\}$. All of the agents are distributed on a network topology. To shorten the communication time among the agents, the task should be allocated to the agents that have the minimum communication distances. The example can be seen in Fig. 1.

2.2. Task allocation algorithm

To describe the relation between agent and the capability in the system, we can give the following definitions:

Definition 1. Agent-capability mapping matrix: $AC = [ac_{ij}]$, where $ac_{ij} = 1$ denotes that agent i has the capability j.

To describe the agent distribution, i.e. the relation between agent and network node, we provide the definition of agent-node mapping matrix.

Definition 2. Agent-node mapping matrix: $AN = [an_{ij}]$, where $an_{ij} = 1$ denotes that agent i locates on node j.

To execute a task, some agent capabilities shall be required.

Definition 3. Task-capability mapping matrix: $TC = [tc_{ij}]$, where $tc_{ij} = 1$ denotes that the capability j is required by task i.

Finally, a task should be allocated to some agents. We provide the definition of task-agent mapping matrix for describing such relation.

Definition 4. Task-agent mapping matrix: $TA = [ta_{ij}]$, where $ta_{ij} = 1$ denotes that task i is allocated to agent j.

We are mainly interested in the communication cost among agents, so we use the shortest distance among nodes to describe the underlying network topology.

Definition 5. Network topology matrix: $N = [n_{ij}]$, where n_{ij} denotes the shortest distance between node i and j.

To make the task allocation adapt for the dynamic topology, we can make the task allocation according to the above matrixes. In our algorithms, we assume that there is a manager agent who knows all of the matrixes.

Now we realize the ideas in Section 2.1 by the introduction of the above matrixes, shown as follows.

Algorithm 1. Task-allocation (task: t).

Step 1. Get the set of the capabilities needed by task t from the task-capability mapping matrix TC, which can be denoted as $C=\{c_1, c_2,\ldots c_n\}$.

Step 2. For all $c_j \in C$:

Get the set A_j of the agents that has the capability c_j from the agent-capability mapping matrix AC, which is denotes as: $A_j=\{a_j^1, a_j^2, \ldots, a_j^{k_j}\}$, where k_j denotes that there are k_j agents have the capability c_j.

Step 4. Select $\overline{a_1}$ from A_1, $\overline{a_2}$ from A_2,..., $\overline{a_n}$ from A_n, so as to the total communication distances among them is the minimum (in current network topology) according to the agent node mapping matrix AN and the network topology matrix N.

Step 5. Stop. Now the task T can be allocated to $\{\overline{a_1}, \overline{a_2}, \ldots, \overline{a_n}\}$.

In Algorithm 1, Step 4 is very important, which shall make the task allocation adapt for the current network topology and agent distribution. We can express it by a *constraint satisfaction problem (CSP)* (e.g. Kumar, 1992; Liu, Jing, & Tang, 2002), shown as follows.

Let X_i be a finite set of variables, $X_i=\{\overline{a_i^1}, \overline{a_i^2}, \ldots, \overline{a_i^n}\}$, X is the set of all X_i, $X=A_1\times A_2\times\ldots\times A_n$ (A_j is the set of the agents that have the capability c_j). Let A be a domain set, containing a finite and discrete domain for each variable:

$$A=\{A_1, A_2, \ldots, A_n\}, \quad \forall j\in[1,n], \overline{a_i^j}\in A_j,$$

Therefore, we should find X_i, such that $\forall(X_j\in X, j\neq i)$, the communication distances in X_j is more than the ones of X_i.

Let a task need n capabilities, and there are k_1 agents that have capability c_1, $A_1=\{a_1^{i_1}|1\le i_1\le k_1\}$; k_2 agents that have capability c_2, $A_2=\{a_2^{i_2}|1\le i_2\le k_2\}$; ...and k_n agents that have capability c_n, $A_n=\{a^{i_n}|1\le i_n\le k_n\}$. Step 4 in Algorithm 1 can be realized by Algorithm 2.

Algorithm 2. Agents voting

```
Step 1  Let min_com_cost=maxnumber; allocated_agen-
        t_set={ };
Step 2  for (int i1=1; i1<=k1; i1++)
            for (int i2=1; i2<=k2; i2++)
              ⋮

                for (int in=1; in<=kn; in++)
                    if com_cost (a1^i1, a2^i2, ..., an^in) <min_com_
                    cost
                    {min_com_cost= com_cost (a1^i1, a2^i2, ...,
                    an^in);
                    allocated_agent_set ={a1^i1, a2^i2, ..., an^in}
                }
Step 3  Output (allocated_agent_set);
```

The function *com_cost* is used to compute the total communication distance among the selected agents, which is shown as Algorithm 3. In Algorithm 3, *an* denotes the element of agent-node mapping matrix AN, n denotes the element of network topology matrix N.

Algorithm 3. com_cost(agents set A).

```
        /*n1 denotes the number of agents in A, n2 denotes
        the number of network nodes */
Step 1  Let total_com_cost=0; number=0; ξ[n1]=0
Step 2  for (int i=1; i<=n1; i++)
            for (int j=1; j<= n2; j++)
            if anij=1 then {ξ(i)=j; number++};
Step 3  for (int i=1; i<=number; i++)
            for (int j=1; j<= number; j++)
            total_comm_cost=total_comm._cost+nξ(i)ξ(j);
Step 4  Return (total_comm._cost);
```

From Algorithm 2, we can see that the time complexity is too high, which is $O(k_1{*}k_2{*},\ldots,{*}k_n)$. Therefore, if the number of capabilities required by a task or the agents set is too large, then the computation cost of the algorithm will be too high. So, now we design an improved algorithm, shown as Algorithm 4, where C denotes the set of capabilities required by the task, A denotes the set of agents that have the capabilities in C. In Algorithm 4, we select the agent that has the more number of capabilities.

Algorithm 4. Improved algorithm for agent voting

```
Step 1  Let allocated_agent_set={ };
Step 2  For each agent ak in the A, containerk=0;
Step 3  For each ci in C
            For each agent aj that have ci,
                If acji=1, then containerj++;
Step 4  Select the agent with the highest container, which is
        denoted as agentr.
Step 5  Let Cr= the set of capabilities that agentr has;
        C=C-Cr;
        A=A-agentr;
        allocated_agent_set=allocated_agent_set ∪ agentr;
Step 6  Repeat Step 2–5, until C is empty;
Step 7  Output (allocated_agent_set).
```

In Algorithm 4, we are prone to select the agents that have more required capabilities which can make the agent number for allocation be lessened. Therefore, the communication cost can also be reduced accordingly. Let a task needs n capabilities, and there are k_1 agents that have capability c_1, k_2 agents that have capability c_2,...,and k_n agents that have capability c_n. Obviously, the complexity of Algorithm 4 is $O(n{*}(k_1+k_2+\ldots+k_n))$, which is lower than the one of Algorithm 2.

3. Resource negotiation for dynamic topology networks

Each agent has different resources, and each resource may be used by different task. We can use a vector to denote the relation between resources and tasks, show as follows:

$$R_i = \{<r_{i1}, t_1>, <r_{i2}, t_2>, \ldots, <r_{ik}, \phi>\}$$

where r_{i1} is used by t_1, r_{i2} is used by t_2,…, and resource r_{ik} is free.

If an agent lacks the necessary resources for executing task, then it may borrow the resources from other agents. Let agent i want to borrow resource r from other agents and there are several resource r_s in the system, and the several resource r_s are owned by different agents and located on different nodes. Therefore, agent i should select r from a nearest agent according to the current network topology. Otherwise, if agent i wants to borrow r from agent j, then some criterions should be satisfied, such as the comparison between their executing tasks' priorities.

Similar to Section 2.1, there are other two binary relations about the resource in the agent systems:

(1) $\rho_5 \subset A \times R$ such that for any $a_i \in A$, $r_j \in R$, $a_i \rho_5 r_j$ holds iff agent a_i owns the resource r_j;
(2) $\rho_6 \subset A \times R$ such that for any $t_i \in T$, $r_j \in R$, $t_i \rho_7 r_j$ holds iff the execution of t_i requires resource r_j;

Next we shall use two kinds of matrixes to describe those two relations.

We can use the agent-resource mapping matrix to describe the relation between agent and resource.

Definition 6. Agent-resource mapping matrix: $AR = [ar_{ij}]$, where $ar_{ij} = 1$ denotes that agent i has resources j.

The execution of task requires some resources, so we provide the definition of *task-resource mapping matrix*.

Definition 7. Task-resource mapping matrix: $TR = [tr_{ij}]$, where $tr_{ij} = 1$ denotes that task i needs the resource j.

Now we can design the agent resource negotiation, shown as Algorithm 5. In Algorithm 5, the N denotes the networking topology matrix, shown as Definition 5; TA is the task-agent mapping matrix, shown as Definition 4.

Algorithm 5. Agent resource negotiation (agent i).

Step 1 From TA, get the tasks set T_i that assigned to agent i;
Step 2 From TR, get the resources set R_i needed by T_i;
Step 3 From AR, get the resources set R_i' that agent i has;
Step 4 If not $R_i' \supseteq R_i$, then go to Step 5, else stop;
Step 5 $\mathbb{R} = R_i - R_i'$;
Step 6 From AR, get the agents set A that has the resources set $\mathbb{R}$;
Step 7 From RN and N, select the agent set A_r that is nearest to agent i and can lend resource to i according to some criterions.
Step 8 agent i borrows resources from A_r.
Step 9 Go to step 4.

Step 7 can be described as follows:

(1) $A_r = \phi$; $R'' = \phi$;
(2) From A, select the nearest agent a_n;
if a_n can lend some resources (R_l) to a_i
then $\{A = A - \{a_n\}$;
$A_r = A_r \cup \{a_n\}$;
$R'' = R'' \cup R_l\}$
(3) If $R'' \supseteq R$ then stop and return A_r;
else goto 2).

Then, how can a_n lend some resources to a_i, and which resource can a_n lend to a_i? Which can be decided according to the task priority between a_n and a_i.

Each agent may have several resources, and each resource may be executing a task.

For example, let a_n has some resource, shown as $\{<r_{n1}, t_{n1}>, <r_{n2}, t_{n2}>, <r_{n3}, \Phi>\}$, which denotes that r_{n1} is executing t_{n1}, r_{n2} is executing t_{n2}, and r_{n3} is free.

Now, let a_i wants to borrow r_{n1} and r_{n3} from a_n. Obviously, r_{n3} can be borrowed immediately since it is free; However, r_{n1} cannot be borrowed immediately since it is executing t_{n1}. At first, we can compute the priority of the task that a_n executes and t_{n1}. If the task that a_n executes is higher than the one of t_{n1}, then a_i can borrow the resource from a_n.

4. Autonomous resource negotiation by knowledge integration

In Section 3, we provide the algorithms for resource negotiation for dynamic topology networks. Through the algorithms in Section 3, we can select the geographical-nearest agents for resource negotiation, therefore, the communication cost among agents can be minimized according to the factual network topology.

However, the algorithms in Section 3 can only be executed by a central controlled fashion, and, the agent cannot negotiate with each other directly.

Aiming at those situations, now we propose an autonomous resource negotiation model. In this model, we use a data structure to represent the agent's knowledge about resource distribution, and agents can integrate individual knowledge for acquiring an enhanced knowledge of the whole resource environment.

In our model, if an agent wants to borrow some resources from other agents, at first it should find the information from the resource knowledge within itself. If it cannot find the information within itself, then we can adopt the algorithms in Section 3.

However, each agent should know the change of network topology. Therefore, in our autonomous negotiation mechanism, there is also a management station in the system

which monitors the network topology and told the agents about the current topology.

4.1. Knowledge representation of resource

We now provide the knowledge representation about resource of agents, and describe how to integrate individual agent knowledge for acquiring an enhanced knowledge of the whole resource environment.

At the initial phase of the system, each agent only has the knowledge about its own resources. Therefore, to know the global knowledge about resources, agents should integrate their knowledge periodically.

We can use the form of *linked list* (Ford & Topp, 1996) to represent the agent's knowledge about resource. The independent items in the linked list are called *rnodes*, each rnode includes four data fields and a pointer indicating the 'next' item in the list. The rnode structure is shown as Fig. 2, where *name* denotes the name of resource, *location* denotes the network node where the resource locates, *ownership* denotes the agent that the resource is owned by, *executing task* denotes the task that the resource is used by.

In the list within an agent, the rnodes are ordered by the distance of the resource from the agent, and the nearest resource is at the head of the list. Therefore, if an agent wants to borrow resource from other agents, it can select the first rnode in the list that owns the resource and the priority of executing task is lower than the agent's task.

For example, Fig. 3(a) shows the resource knowledge of agent a_1, and Fig. 3(b) shows the resource knowledge of agent a_2.

If a_1 lacks resource r_1 while it executes its task, it shall search the resource information in its resource knowledge. In its linked list, there are two r_1. The r_1 owned by a_3 is the nearest and the priority of t_5 is lower than the priority of the task executed by a_1, so a_1 can borrow r_1 from a_3.

If a_2 lacks resource r_1 while it executes its task, it at first searches its linked list. However, there is no information about r_1 in its linked list. Therefore, now we should adopt the algorithms in Section 3, i.e. the central controlled fashion.

```
Typedef struct node{
elementtype1 name;
elementtype2 location;
elementtype3 ownership;
elementtype4 executingtask;
struct node *next;
} rnode;
rnode resource[n];
```

Fig. 2. The rnode structure.

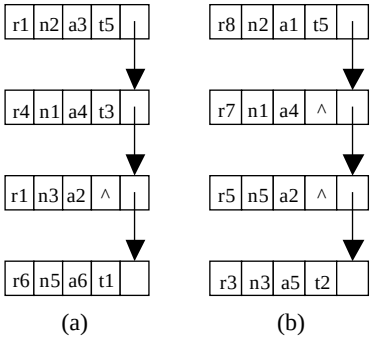

Fig. 3. A example of agent resource knowledge.

4.2. Integration of agent resource knowledge

As said above, in the initial phase of the multi-agent system, each agent only has the knowledge about its own resources. To achieve the global resource distribution information, agents need to integrate their resource knowledge. The integration of resource knowledge can be realized under the way that each agent broadcasts its resource knowledge to the agents on geographical neighboring nodes periodically. Then the agent can integrate the resource knowledge and arrange the linked list according to its information about the network topology from the management station. As the time going, agents can achieve the global information about resources step by step.

Therefore, the knowledge integration about resources of the two agents is shown in Fig. 4. Fig. 4(a) is the integrated resource knowledge in a_1, Fig. 4(b) is the one in a_2. In Fig. 4(a) or (b), the rnodes is arranged according to the distance of the resource from a_1 (or a_2), the rnode for the nearest resource is at the head of the linked list.

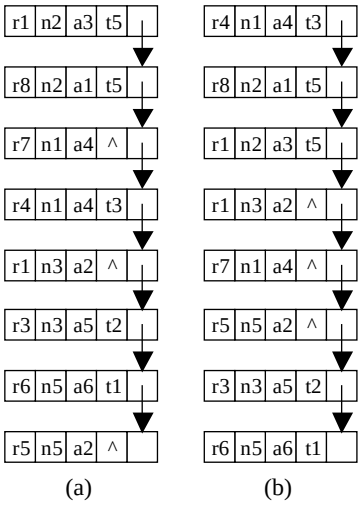

Fig. 4. The resource knowledge after integration.

In the integration of resource knowledge, we mainly adopt the *inserting* operation. Therefore, the time complexity of the integration is $O(n)$.

Each time the network topology is changed, the management station can probe the current topology and broadcast the topology information to all agents, and the agents then re-arranged their linked lists.

5. Case studies and performance analyses

5.1. A case

To simulate the dynamic network topology, we can consider a $M \times N$ grid, and the edges in the grid can always change. The communication among agents can only go along the horizontal or vertical directions. We can realize our multi-agent coordination model in these case studies, and make analyses for the cases, so as to testify the effectiveness of our model.

Fig. 5 is the simulated agent system environment. Let there is a 30×30 grid, and the distance of all edges are the same. We can assume the distance of each edge is d. And the communication cost is the function of d. The more d is, the more the communication cost is. We put some agents having different capabilities into the grid randomly.

5.2. Task allocation

From Fig. 5, the agent-capability mapping matrix is shown as (1):

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{1}$$

The dimension of agent-node mapping matrix of the agent system in Fig. 5 is too large, which is 12×90, so now we do not list the whole matrix for saving space, but only list some elements for the agent location, show as (2).

$$\begin{aligned} &an_{7,43} = 1 \quad an_{11,93} = 1 \quad an_{5,203} = 1 \quad an_{1,218} = 1 \\ &an_{6,341} = 1 \quad an_{2,347} = 1 \quad an_{9,395} = 1 \quad an_{3,557} = 1 \\ &an_{4,577} = 1 \quad an_{8,738} = 1 \quad an_{10,779} = 1 \quad an_{12,790} = 1 \end{aligned} \tag{2}$$

The network topology matrix $N=[n_{ij}]$, if the location of node i is denoted by coordinates (m_1,n_1) in the grid, and the location of node j is denoted by (m_2,n_2) in the grid, then there is:

$$n_{ij} = |m_2 - m_1| + |n_2 - n_1| - 1 \tag{3}$$

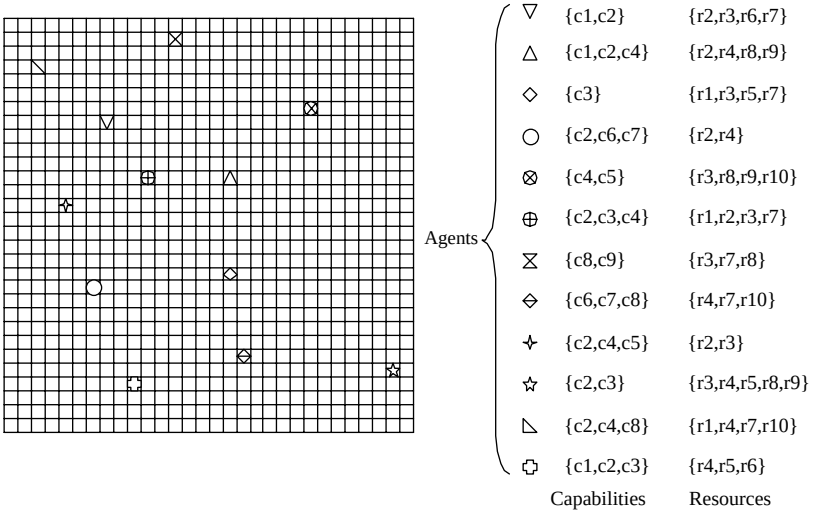

Fig. 5. A simulated agent system environment.

Now let there are 10 tasks that the system shall execute, and each task needs some capabilities, and the task-capability mapping matrix is shown as (4).

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{4}$$

Now we consider the task allocation for task t_5. According to (4), we can see that t_5 needs the capabilities set $\{c_2,c_4,c_6,c_8\}$. From (1), the agents sets that respectively has the capability c_2, c_4, c_6, c_8 are A_2, A_4,A_6,A_8.

$$A_2=\{a_1,a_2,a_4,a_6,a_9,a_{10},a_{11},a_{12}\} \quad A_4=\{a_2,a_5,a_6,a_9,a_{11}\}$$
$$A_6=\{a_4,a_8\} \qquad\qquad A_8=\{a_7,a_8,a_{11}\} \tag{5}$$

Now we can use Algorithm 2 to make the task allocation, then the agents set of task allocation is:

$\{a_2,a_8\}$. Scheme (1)

If we use Algorithm 4 to make the task allocation, then the agents set of task allocation is:

$\{a_{11},a_4\}$. Scheme (2)

Otherwise, we also give some random task allocation schemes, shown in Table 1. Let the agents in the task execution implement fully mutual communications, and then we can compute the communication cost of the schemes. We can assume the communication cost of each step in the grid is d.

According to (2) and (3), the shortest distance between a_6 and a_8 is 14, so the communication cost of Scheme (1) is $14d$; the shortest distance between a_{11} and a_4 is 19, so the communication cost of Scheme (2) is also $19d$.

Table 1
Performance comparison among different Schemes (1)

Scheme	Com_cost	Scheme	Com_cost
$\{a_2,a_8\}$	14d	$\{a_6,a_8\}$	19d
$\{a_{11},a_4\}$	19d	$\{a_9,a_8\}$	24d
$\{a_1,a_2,a_4,a_7\}$	87d	$\{a_4,a_7,a_5\}$	65d
$\{a_2,a_5,a_8,a_7\}$	100d	$\{a_8,a_{11}\}$	36d
$\{a_2,a_5,a_8\}$	45d	$\{a_4,a_7,a_9\}$	48d

Under the same way, we can compute the communication costs of other random schemes, the results are shown in Table 1. For example, the fully mutual communications of the scheme $\{a_2, a_5, a_8\}$ are $\{a_2 \leftrightarrow a_5, a_2 \leftrightarrow a_8, a_5 \leftrightarrow a_8\}$. The shortest distance between a_2 and a_5 is $10d$, the shortest distance between a_5 and a_8 is $13d$, and the shortest distance between a_2 and a_8 is $22d$. Therefore, the communication cost of scheme $\{a_2, a_5, a_8\}$ is $45d$.

From Table 1, we can see that the communication costs of Scheme (1), (2) are lower than the ones of other random schemes. Therefore, our model can produce the optimal task allocation scheme.

5.3. Resource negotiation

Now we take an example to illustrate the process of resource negotiation. We consider the optimal task allocation scheme in Section 5.2, i.e. {a2, a8}.

Let (6) be an agent-resource mapping matrix in the system, and (7) is the task-resource mapping matrix.

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{6}$$

$$\begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \end{pmatrix} \tag{7}$$

From (7), we can see that task t_5 needs the resources $\{r_1, r_2, r_3, r_5, r_6, r_7, r_8, r_9, r_{10}\}$. However, from (6), we can see that agent a_2 and a_8 only have the resources $\{r_2, r_4, r_7, r_8, r_9,$

$r_{10}\}$. Therefore, they shall make resource negotiation with other agents to borrow the resources $\{r_1, r_3, r_5, r_6\}$.

Now we let the task-agent mapping matrix be shown as (8), and we also assume the priorities of the tasks is as follows: $t_1 \succ t_2 \succ t_3 \succ t_4 \succ t_5 \succ t_6 \succ t_7 \succ t_8 \succ t_9 \succ t_{10}$.

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \quad (8)$$

The agents set that have the resources $\{r_1, r_3, r_5, r_6\}$ is $\{a_1, a_3, a_5, a_6, a_7, a_9, a_{10}, a_{11}, a_{12}\}$, which can be called as $\ddot{A}$.

Now we can define the distance between an agent a_j and $\ddot{A}$, shown as (9).

$$d(\ddot{A}, a_j) = \min_{a_i \in \ddot{A}}[d(a_i, a_j)] \quad (9)$$

where $d(x_i,x_j)$ denotes the shortest path distance between x_i and x_j.

According to the agent-node mapping matrix and the network topology, the agents set $\ddot{A}$ can be arranged according to the distances to $\{a_2, a_8\}$ (computed according to (9)), shown as $\{a_6, a_3, a_{12}, a_5, a_{10}, a_1, a_7, a_9, a_{11}\}$.

According to (8), a_6 is allocated with t_1, t_3, and t_7. The priority of t_1 is higher than t_5, therefore, a_6 cannot lend the resources to a_2 and a_8. The priority of the tasks allocated to a_3 is lower than t_5, therefore, a_3 can lend the resources $\{r_1, r_3, r_5\}$.

Now, $\{a_2, a_8\}$ only need to borrow r_6 from other agents. The agents set that has the resource r_6 can be arranged according to their distances to $\{a_2, a_8\}$ as $\{a_{12}, a_1\}$. The task that is allocated to a_{12} is t_9 whose priority is lower than the one of t_5, therefore, a_{12} can lend the resource r_6 to a_2 and a_8.

Therefore, a_2 and a_8 can borrow the resources $\{r_1, r_3, r_5, r_6\}$ from $\{a_3, a_{12}\}$. Obviously, the communication cost for resource negotiation of this scheme is the minimum. Therefore, our algorithms can achieve the optimal result.

As far as the autonomous resource negotiation mechanism in Section 4, it selects the nearest resource for negotiation in the linked list. Therefore, the communication cost is sure to be the minimum. For saving space, here we do not make case studies for the autonomous negotiation mechanism.

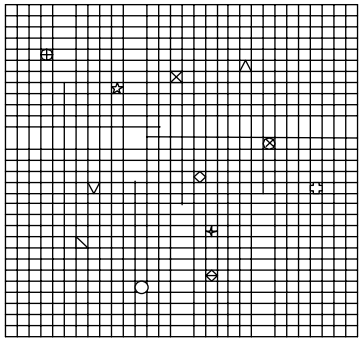

Fig. 6. Network topology and agent distribution (2).

5.4. Performance analyses of task allocation for the changed network topologies and agent distributions

Now we analyze the performance of our task allocation for the changed network topologies and agent distributions. Since the analysis of the performance of resource negotiation is similar to the one of task allocation, so here we overlook it.

Now we change the topology of the grid and the agent distribution, and test our algorithms of task allocation. The grid topology and the agent distribution are shown as Fig. 6.

Now, we assume t_8 arrives to the system, according to Algorithm 2 in Section 2.2, the agents set of task allocation is $\{a_1,a_{11},a_4\}$; according to Algorithm 4, the agents set of task allocation is $\{a_2,a_4\}$. Now we can make some random task allocation schemes, shown in Table 2. Then, we compare the communication costs between the constructed task allocation schemes and other random task allocation schemes, shown as Table 2.

Table 2
Performance comparison among different schemes (2)

Scheme	Com_cost	Scheme	Com_cost
$\{a_1,a_{11},a_4\}$	25d	$\{a_{12},a_9,a_8\}$	32d
$\{a_2,a_4\}$	28d	$\{a_1,a_5,a_4\}$	53d
$\{a_2,a_8\}$	31d	$\{a_2,a_9,a_8\}$	60d
$\{a_8,a_9,a_{12}\}$	31d	$\{a_{12},a_9,a_4\}$	51d
$\{a_1,a_2,a_4\}$	63d	$\{a_2,a_9,a_4\}$	71d

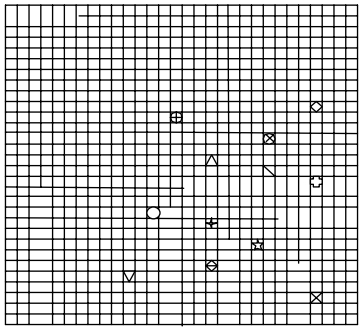

Fig. 7. Network topology and agent distribution (3).

Table 3
Performance comparison among different schemes (3)

Scheme	Com_cost	Scheme	Com_cost
$\{a_2,a_8\}$	15d	$\{a_2,a_4,a_{11}\}$	27d
$\{a_1,a_8\}$	19d	$\{a_1,a_4,a_8\}$	37d
$\{a_{12},a_8\}$	24d	$\{a_1,a_4,a_7\}$	61d
$\{a_2,a_4,a_8\}$	35d	$\{a_1,a_4,a_{11}\}$	41d
$\{a_2,a_4,a_7\}$	58d	$\{a_{12},a_4,a_{11}\}$	35d

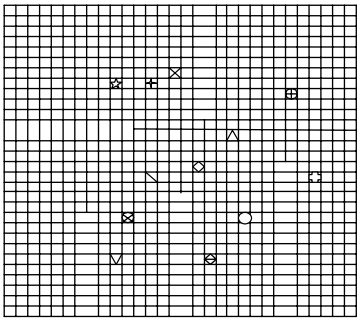

Fig. 8. Network topology and agent distribution (4).

Now, we change the network topology and agent distribution, shown as Fig. 7. We assume t_4 arrives to the system, according to Algorithm 2 in Section 2.2, the agents set of task allocation is $\{a_2,a_8\}$; according to Algorithm 4, the agents set of task allocation is $\{a_1,a_8\}$. Now we can compare the communication costs between the constructed task allocation schemes and other random task allocation schemes, shown as Table 3.

Now, we change the network topology and agent distribution, shown as Fig. 8. We assume t_{10} arrives to the system, according to Algorithm 2 in Section 2.2, the agents set of task allocation is $\{a_2\}$; according to Algorithm 4, the agents set of task allocation is also $\{a_2\}$. Now we can compare the communication costs between the constructed task allocation schemes and other random task allocation schemes, shown as Table 4.

Now, we change the network topology and agent distribution, shown as Fig. 9. We assume t_1 arrives to the system, according to Algorithm 2 in Section 2.2, the agents set of task allocation is $\{a_{12},a_7\}$; according to Algorithm 4, the agents set of task allocation is $\{a_1,a_7\}$. Now we can compare the communication costs between the constructed task allocation schemes and other random task allocation schemes, shown as Table 5.

Table 4
Performance comparison among different schemes (4)

Scheme	Com_cost	Scheme	Com_cost
$\{a_2\}$	0	$\{a_1,a_5\}$	4d
$\{a_1,a_6\}$	30d	$\{a_1,a_9\}$	19d
$\{a_1,a_{11}\}$	10d	$\{a_{12},a_5\}$	23d
$\{a_{12},a_6\}$	9d	$\{a_{12},a_9\}$	21d
$\{a_{12},a_{11}\}$	16d		

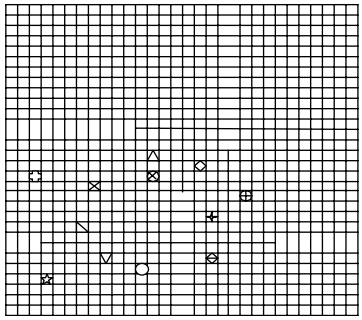

Fig. 9. Network topology and agent distribution (5).

Table 5
Performance comparison among different schemes (5)

Scheme	Com_cost	Scheme	Com_cost
$\{a_{12},a_7\}$	5d	$\{a_2,a_8,a_7\}$	37d
$\{a_1,a_7\}$	7d	$\{a_2,a_{11},a_7\}$	23d
$\{a_2,a_7\}$	7d	$\{a_{12},a_8,a_7\}$	43d
$\{a_1,a_8,a_7\}$	45d	$\{a_{12},a_{11},a_7\}$	17d
$\{a_1,a_{11},a_7\}$	15d		

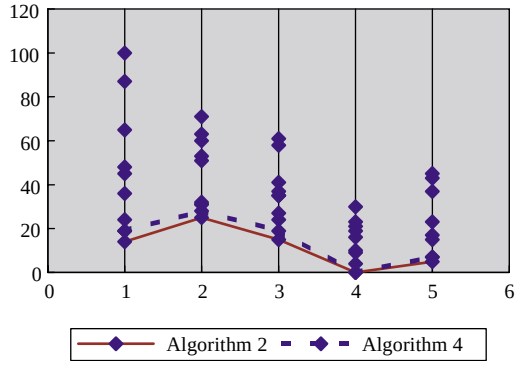

Fig. 10. Summary of the performance comparison.

Now we combine the results of all of above cases, shown as Fig. 10. In Fig. 10, the *Y*-coordinate denotes the communication cost, and the *X*-coordinate denotes the number of cases. From Fig. 10, we can see that the communication costs of the schemes produced by our algorithms are the minimum. Therefore, our model can produce the optimal task allocation schemes.

6. Conclusion

Coordination mechanism is widely explored in multi-agent systems. Especially, the task allocation and resource negotiation are the key issues in the research of agent coordination.

Nowadays, the topologies of networks are always changed during their operations, therefore, the multi-agent systems on such networks shall adapt themselves for

the network topology. However, the existing multi-agent coordination mechanisms do not have such adaptation ability. In this paper, we explore the agent task allocation and resource negotiation, and provide some algorithms to make the agents coordinate themselves for the network topology. In the provided algorithms, the factors of network topology and agent distribution are considered. The algorithms can realize the effective task allocation and resource negotiation according to the current network topology, and the optimal result for agent communication cost can be achieved.

At last, we make some case studies, and make analyses for the algorithms when the network topology and agents distribution are changed. From the case studies and the performance analyses, we conclude that our algorithms can perform well in the dynamic networks.

References

Buccaafurri, F., Rosaci, D., Sarne, G. M. L., & Palopoli, L. (2004). Modeling cooperation in multi-agent communities. *Cognitive Systems Research*, *5*, 171–190.

Kai-Hsiung Chang, Suebskul Phiphobmongkol, & William B. Day. (1993). An agent-oriented multiagent planning system. *Proceedings of the 1993 ACM conference on computer science* (pp. 107–114). Indianapolis, Indiana, United States.

Excelente-Toledo, Cora Beatriz, & Jennings, Nicholas R. (2004). The dynamic selection of coordination mechanisms. *Autonomous Agents and Multi-Agent Systems*, *9.55-85*.

Findler, Nicholas V. (1995). Multiagent coordination and cooperation in a distributed dynamic environment with limited resources. *Artificial Intelligence in Engineering*, *9*(3), 229–238.

Ford, William, & Topp, William (1996). *Data structures with C++*. Prentice Hill, Inc. pp. 383–474.

Georgeff, M. (1983). Communication and interaction in multiagent planning. In: *Proceedings of national conference on artificial intelligence* (pp. 125–129). Washington, DC.

Gruber, T. (1991). The role of common ontology in achieving sharable, reusable knowledge bases. In J. A. Allen, R. Fikes, & E. Sandewall (Eds.), *Principles of knowledge representation and reasoning: proceedings of the second international conference* (pp. 601–602). Cambridge, MA: Morgan Kauffman, 601–602.

Kai-Hsiung Chang, & William B. Day. (1990). Adaptive multiagent planning in a distributed environment. *Proceedings of the third international conference on Industrial and engineering applications of artificial intelligence and expert systems* (Vol. 2) (pp. 828–837). Charleston, South Carolina, United States.

Kraus, S. (1997). Negotiation and cooperation in multi-agent environment. *Artificial Intelligence*, *94*, 79–97.

Kraus, Sarit, & Plotkin, Tatjana (2000). Algorithms of distributed task allocation for cooperative agents. *Theoretical Computer Science*, *242*, 1–27.

Kumar, V. (1992). Algorithm for constraint satisfaction problem: A survey. *AI Magazine*, *13*(1), 32–44.

Liu, Jiming, Jing, Han, & Tang, Y. Y. (2002). Multi-agent oriented constraint satisfaction. *Artificial Intelligence*, *136*, 101–104.

Palmer, D., Kirschenbaum, M., Murton, J., & Zajac, K. (2003). Decentralized cooperative auction for multiple agent task allocation using synchronized random number generators. *Proceedings of the IEEE/RSJ, international conference on intelligent robots and systems, Las Vegas, Nevada* .

Tumas Sandholm. (1993). An implementation of the contract net protocol based on marginal cost calculations. *Proceedings of the 12th international workshop on distributed artificial intelligence* (pp. 295–308). Hidden Valley, Pennsylvania.

Schuster, Stefan (2000). Knowledge representation and graph transformation. In H. Ehrig, et al. (Ed.), *Graph transformation, LNCS 1764* (pp. 228–237). Berlin: Springer, 228–237.

Shehory, Onn, & Kraus, Sarit (1998). Task allocation via coalition formation among autonomous agents. *Artificial Intelligence*, *101*, 165–200.

Smith, R. G. (1980). The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12), 1104–1113.

Tanaev, V. S., Sotskov, Y. N., & Strusevich, V. A. (1994). *Scheduling theory, multi-stage systems*. Dordrecht, Netherlands: Kluwer.