Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 107246, 12 pages
http://dx.doi.org/10.1155/2014/107246

Research Article

Reorganizing Complex Network to Improve Large-Scale Multiagent Teamwork

Yang Xu, Pengfei Liu, and Xiang Li

School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China

Correspondence should be addressed to Yang Xu; xuyang@uestc.edu.cn

Received 5 February 2014; Accepted 26 February 2014; Published 14 April 2014

Academic Editor: Guoqiang Hu

Large-scale multiagent teamwork has been popular in various domains. Similar to human society infrastructure, agents only coordinate with some of the others, with a peer-to-peer complex network structure. Their organization has been proven as a key factor to influence their performance. To expedite team performance, we have analyzed that there are three key factors. First, complex network effects may be able to promote team performance. Second, coordination interactions coming from their sources are always trying to be routed to capable agents. Although they could be transferred across the network via different paths, their sources and sinks depend on the intrinsic nature of the team which is irrelevant to the network connections. In addition, the agents involved in the same plan often form a subteam and communicate with each other more frequently. Therefore, if the interactions between agents can be statistically recorded, we are able to set up an integrated network adjustment algorithm by combining the three key factors. Based on our abstracted teamwork simulations and the coordination statistics, we implemented the adaptive reorganization algorithm. The experimental results briefly support our design that the reorganized network is more capable of coordinating heterogeneous agents.

1. Introduction

Cooperative multiagent and multirobot teams are promising in the domain of distributed artificial intelligence, such as controlling of a large number of unmanned aerial vehicles [1], coordinating soldiers, agents and robots in battlefield [2], and searching and rescuing in disasters [3]. In those teams, agents work together toward their common goal, and similar to the infrastructure of human society, they only closely coordinate with a few teammates either logically or under the constraints of physical communications, with a peer-to-peer associate network structure. For example, in a multi-UAV team, members communicate through a wireless ad hoc network structure. When the team scales up, the network presents the complex network structure in the dynamic team coordination [4].

How to improve the team performance by adjusting the network topology has been extensively studied. Some special complex networks attributes are found to be important to the social network performance. For example, Gaston found that the tight coupled network structure will weaken the team performance because of the excessive consumption [5]. Scerri has analyzed how the network structure affects the efficiency of the multiagent communication network [6]. Following those discoveries, structure-oriented approaches [7] take the advantages of the discoveries of complex network effects, such as the scale-free, small-world property and the community structure on the team performance [8, 9]. On the other hand, actor-oriented approaches [7] focus on analyzing the characters of agents' behaviors on the networked coordination and making use of agents' role features to adjust team organization based on a given network structure such as hierarchical or grid-based networks [7]. For example, Jiang et al. [10] proposed a task assignment algorithm for heterogeneous agents with an understanding of its "actor" in the social network. Although significant progress has been made in both structure and actor-oriented approaches, no previous research combines both advantages to build an integrated team reorganization algorithm in real multiagent coordination domains [7].

In this paper, to combine both structure and actor-oriented approaches, we made three efforts. First, we analyze

how different network topologies with different complex attributes may influence the agents' networked coordination by Markov chain model. The discovery is exciting because potential scale-free and small-world effects help the team a lot especially when coordination interaction can be intelligently routed. Second, we found that agents always try to connect each other directly or within very short distance if they are closely cooperated. However, the cooperative relationship is the intrinsic character of the team and independent of how team is organized, for example, the capabilities of heterogeneous agents and the subteams in which agents closely cooperate for joint activities [11]. Similar to human society, people closely coordinate or communicate because some kinds of relationships exist. For example, a letter will be delivered from the mother to her daughter, or bread will be sent from the bakery to a hungry person. Human society will keep routing between those pairs of persons no matter how far they are. Therefore, if we can reorganize the team according to this nature and connect those closely coordinated agents with less links, the team will be benefited most.

To make our team organization optimization approach possible, we simulated the peer-to-peer coordination of large-scale heterogeneous multiagent systems. In our simulation, the resources, tasks, and other coordination information are abstracted and encapsulated into messages, which can be passed from the source agent to the destination via the coordination network. This method is easy to track the messages so that we can infer which pairs of agents are closely coordinated. To find the closely cooperative relationships, we built a connection matrix and recorded messages' resources and destinations. In addition, the subteam model is used to record the agents who are involved in the same joint activity.

As the last part of our design, we set up the adaptive reorganization algorithm to integrate the three key factors we have analyzed. In the network construction phase, by building a probability model for each pair of agents, agents recursively pick their neighbors based on their probabilities. The probability of connecting each pair of agents is measured by how closely cooperated they are, within a subteam, or how much they help to promote the scale-free and small-world effects if they are directly connected. Therefore, in our algorithm, agents always pick neighbors who are closely cooperative, with higher degrees and shorter distances, but with different important factors. Our experiment results briefly match our expectations. By reorganizing the team as a small-world and scale-free network and connecting closely cooperative agents with shorter distances, the network can significantly expedite team performance.

2. Scalable Team Coordination

In this section, we build the scalable multiagent team network model, in which pairs of agents are defined as connected only when they are able to interact with each other directly and each agent only maintains its peer-to-peer connection with a few of others. The objective of their interactions is to jointly perform their tasks so that their complex team goal can be achieved.

2.1. Large Team Organization. The organizational topology is described as an undirected graph: $G = (A, E)$, where multiagent A is the node set of G, $A = \{a_1, \ldots, a_k\}$ and E is the set of edges; if there is $e_k = \langle i, j \rangle$, then coordination messages could be transmitted between a_i and a_j directly; that is, they are neighbors to each other. $N(i)$ is defined as the set of a_i's neighbors.

G could be organized based on the properties of complex networks. In this paper, we are mainly interested in four of the topologies: random network, grid network, small-world network, and scale-free network. Preliminary studies [12] found that each topology encodes the following different fundamental properties.

(i) Degree: the degree of agent a_i is $d(i) = |n(i)|$.

(ii) Average degree: $\overline{d} = (1/N) \sum_{i \in V} |n(i)|$ is the average number of neighbors of all agents, for any complex network $\overline{d} \ll |v|$.

(iii) Degree distribution: $p(k) = \Pr[d = k]$ is defined as a fraction of agents (the number of such agents is d) with the degree k.

(iv) Distance: $distance(i, j)$ defines the least number of hops to communicate between agents a_i and a_j. Specifically, $distance(i, j) = 1$, if $\langle i, j \rangle \in E$.

(v) Average distance is the average distance between any pair of agents:

$$l = \frac{1}{N(N-1)} \sum_{\forall a_i, a_j \in A} distance(i, j). \quad (1)$$

Different complex network topologies can be described according to the properties. In this paper, we mainly consider the two effects and used *degree distribution* to express the scale-free effect and *average distance* to express the small-world effect.

2.2. Multiagent Team Coordination. A large multiagent team coordination can be briefly described as follows: agents are cooperative on a joint goal. It can be decomposed into discrete subgoals $g_1, \ldots, g_i$. To achieve the subgoals, the corresponding tasks $\alpha_1, \ldots, \alpha_i$ are typically performed by individuals. Agents must perform the individual tasks α, when they are applicable, for the team to receive reward. An amount of reward will be received by the team when an agent performs a task. The reward depends on the agent and task, the capability of the agent, and the resources that the agent has. Specifically,

$$Reward(a, \alpha, Capability(a, \alpha), Holds(a)) \longrightarrow \mathcal{R}. \quad (2)$$

The function $Assigned(a, \alpha) = 1$, if agent a is assigned a task α; otherwise it is equal to 0. A given task is only allowed to be assigned to one agent at any time; that is, $\sum_{a \in A} Assigned(a, \alpha) \leq 1$. However, agents may expect different utilities on the same task based on their capabilities. For example, we should expect higher reward for a fireman to do fire fighting than a nonexperienced civilian. The function

Capability(a, α) projects a real value to denote the expected utility that agent a performs α.

Agents always require sharable resources to perform tasks. These resources, $Res = \{res_1, \ldots, res_m\}$, are discrete and nonconsumable. Agent a has the exclusive access to resources $Holds(a) \subseteq Res$. Only one agent may hold a resource at any time; that is, $\forall a, b \in A$, $a \neq b$, $Holds(a) \cap Holds(b) = \emptyset$.

The teamwork is to maximize the reward for the team, while minimizing the *costs of coordination*. The overall reward is

$$\sum_{i=1}^{n} \sum_{a \in A} Assigned\,(a, \alpha_i)\, Reward \times (a, \alpha_i, Capability\,(a, \alpha)\,, Holds\,(a))\,. \tag{3}$$

The costs of coordination are very general and in some cases hard to be defined. Here we are mainly concerned with the volume of communication.

2.3. Coordination Decision Process. The objective of agents' interactions is to jointly coordinate themselves so that their common goal could be achieved. In large team coordination, similar to human society, agents always forward the incapable tasks and resources across the network. Once an agent accepts a task or resource according to its capability and what it is performing, it will execute the task or make use of the resource. The key of the coordination is to optimize their coordination so that the best capable agents could be reached to as fast so that agents' communication and assignment delay can be minimized.

Specifically, in our abstracted coordination simulator, initiated tasks or resources are encapsulated into messages; each agent executes Algorithm 1, which describes a general way of agents' coordination. Agents firstly check whether new tasks become applicable. If it is, the agent will encapsulate the task into a message and add it into its message queue so that the messages can be processed (lines 3–7). Next, the agent will merge all the messages passed from other agents to its message queue (line 8). It then processes all the messages in the queue. If a message represents a task, the agent will accept the task when its capability to perform that task is higher than message's threshold (lines 11–14); otherwise, the agent will choose a neighbor to pass that message to (line 17). If the message encapsulated a resource and the agent's need for that resource to perform its waiting tasks is higher than message's current threshold [13], this resource will be held; otherwise it is passed to a neighbor (lines 19–27). Note that when a message is sent, the message will be removed from that agent's list. Finally, the agent will check whether any pending tasks can now be executed (line 30) and release any resources from completed tasks (lines 32–37).

3. Coordination Efficiency over Different Network Topologies

In this section, we briefly analyze how team organization can make the team coordination performance different. We model the team coordination messages' routing over the network as a finite Markov chain, which is briefly illustrated as in Figure 1. For a specific message movement, we can define different states. In Figure 1, s_i defines the state that the message moves to an agent with the shortest distance of i to the sink agent. The transition probability $P_{i,j}$ defines that of the message being passed from state i to j. Because there is only one step move for each message at any horizon, $P_{i,j} = 0$ except for $j \in \{i-1, i, i+1\}$. Therefore, for a state $s_i \neq s_0$, the message may move closer to the destination ($P_{i,i-1}$), stay on the same level ($P_{i,i}$), or move far away ($P_{i,i+1}$) as the three statuses shown in Figure 2. When the message reaches state s_0, it will be kept at the destination and $P_{0,0} = 1$. If we suppose that u is the initial probability distribution of the message being in state s, according to the theory of Markov chains [14], the probability that the message reaches the sink agent after n steps can be calculated as

$$P_s^n = u \times P^n. \tag{4}$$

As the agents transmit the messages randomly to anyone of their neighbors, there will be different distances between the source and destination. Figure 2 shows the relative rates of $P(s_i, s_{i-1})$ (marked as "Close"), $P(s_i, s_i)$ (marked as "Same"), and $P(s_i, s_{i+1})$ (marked as "Further") for scale-free and random networks [6]. Notice that we average $P(s_i, s_j)$ over each node at distance i, though this will vary from node to node in different cases. The x-axis shows the distance from a node to the target node, that is, the target agent i. The y-axis shows the proportion of the states of "Further," "Same," and "Closer," and different areas represent the corresponding proportions with a sum of 100%. In general, the closer the agent is to the sink agent, the more likely the random movement is to lead the message further from it. Conversely, the further the agent is from the sink agent, the more likely the random movement is to lead the message closer to it. Figure 2 shows different complex network probability distributions with different messages' movement probabilities.

Figure 2(a) shows the messages' state probability transition matrix P with a typical scale-free network organization:

$$\begin{pmatrix} 0.02 & 0.98 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.01 & 0.1 & 0.89 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.05 & 0.25 & 0.7 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.2 & 0.35 & 0.45 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 0.25 & 0.25 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.75 & 0.1 & 0.15 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.89 & 0.01 & 0.1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}. \tag{5}$$

Figure 2(b) shows that, with a typical random network organization,

$$\begin{pmatrix} 0.01 & 0.99 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.01 & 0.03 & 0.96 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.01 & 0.24 & 0.75 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.15 & 0.5 & 0.35 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.6 & 0.25 & 0.15 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.8 & 0.05 & 0.15 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.88 & 0.02 & 0.1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}. \tag{6}$$

```
(1) ApplicableTasks = [], OwnTasks = [], Holds = [], Messages = [];
(2) while (true) do
(3)   for (α ∈ agent a_i & α ∉ ApplicableTasks) do
(4)     if (Applicable(α)) then
(5)       ApplicableTasks.append(α);
(6)       Messages.append(CreateMessages(α));
(7)     end if
(8)     Messages.append(recvMessages());
(9)   end for
(10)  for (m ∈ Messages) do
(11)    if (m is TaskMessages(α)) then
(12)      if (GetCap(α) > m.threshold) then
(13)        if (α ∉ OwnTasks) then
(14)          OwnTasks.append(α);
(15)        end if
(16)      else
(17)        SendToNeighbour(m);
(18)      end if
(19)    else if (m is ResourceMessages(r)) then
(20)      m.threshold += δ;
(21)      if (GetNeed(r) > m.threshold) then
(22)        if (r ∉ Holds) then
(23)          Holds.append(r);
(24)        end if
(25)      else
(26)        m.threshold −= δ;
(27)        SendToNeighbour(m);
(28)      end if
(29)    end if
(30)    CheckExecution(OwnTasks, Holds);
(31)  end for
(32)  for (α ∈ OwnTasks) do
(33)    if (α is complete) then
(34)      OwnTask.remove(α);
(35)      for (r ∈ ChkUnneed(OwnTask, Holds)) do
(36)        Hold.remove(r);
(37)        SendToNeighbour(CreateMessages(r));
(38)      end for
(39)    end if
(40)  end for
(41) end while
```

Algorithm 1: Coordination decision process.

Suppose that the same message's initial distribution is $u = [0.01\ 0.15\ 0.15\ 0.10\ 0.10\ 0.17\ 0.16\ 0.16]$ and after given steps of message's random movements, the state probability distribution is listed as in Table 1. For example, after 1000 steps, the state probability distribution for a scale-free network is $[0.858\ 0.003\ 0.016\ 0.050\ 0.056\ 0.016\ 0.001\ 0.000]$, where in 86% of cases this message has reached the sink agent. On the other hand, the state probability distribution for a random network after 1000 steps is $[0.653\ 0.003\ 0.020\ 0.106\ 0.182\ 0.036\ 0.000\ 0.000]$, where in only about 65% cases this message has reached the sink agent. The efficiency of information transmission in a scale-free network is significantly higher than in a random network.

Table 1: Random walk in complex networks.

	Steps	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7
Scale-free	100	0.310	0.012	0.08	0.244	0.272	0.078	0.004	0.0
	200	0.421	0.010	0.067	0.204	0.228	0.065	0.004	0.0
	500	0.658	0.006	0.040	0.121	0.135	0.039	0.002	0.0
	1000	0.858	0.003	0.016	0.050	0.056	0.016	0.001	0.0
Random	100	0.248	0.007	0.042	0.229	0.394	0.079	0.001	0.0
	200	0.310	0.006	0.039	0.211	0.362	0.073	0.001	0.0
	500	0.467	0.005	0.030	0.163	0.280	0.056	0.001	0.0
	1000	0.653	0.003	0.020	0.106	0.182	0.036	0.000	0.0

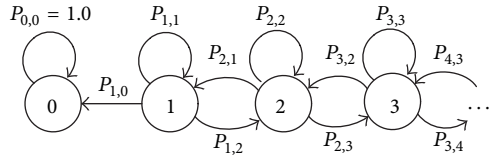


Figure 1: Markov chains model on messages' movement between agents.

4. Team Reorganization Approach

The theoretical analysis demonstrates that the organization structure of large-scale multiagent team does influence its performance efficiency. In this section, we will introduce our team adjusting approach to construct better team organization so that team coordination efficiency could be improved. Taking both the intrinsic characters of agents' coordination and the complex network effects into consideration, we have proposed an integrated adaptive algorithm and built two heuristic models to learn the agents' closely coordinative relationships.

Our previous study has found that communication efficiency of the team, which is measured by the number of hops a message goes across the team, will be improved if messages are forwarded down the link with higher probability towards the destination agent. Therefore, we can build a data structure to learn from the sampled team coordination performance data and transferred into the reorganization algorithm.

The basic process of our reorganization approach is described as in Figure 3. In the learning process, we will sample the interactions between agents from the coordination simulations and build two heuristic models: *connection matrix* and *subteam* to find closely coordinative relationships for each pair of agents. Next, by taking the complex network attributes, *connection matrix* and *subteam* models, into consideration, we build a probability model to measure the connection between each pair of agents. The high probability will be applied for agents who are closely cooperated, within a subteam, or promote the scale-free and small-world effects. Therefore, we can transform the probability model into an adaptive reorganization algorithm, where agents recursively pick their neighbors based on their probabilities.

4.1. Connection Matrix. The coordination between agents is somewhat similar to human society, in which groups of people closely coordinate and communicate according to their common interests. Therefore, tasks or resources always come from their source to the agents who are capable of performing the task or using the resource encapsulated. For example, a piece of information about finding a hostile tank always starts from an unmanned scout good at information gaining and is useful to the UAVs who can attack tanks. This cooperative relationship is an intrinsic character of the team and is independent of how the team is organized or how the coordination is delivered.

If the messages coming from the source can be delivered to their destinations with shorter paths, the coordination is improved. In addition, the pairs of the source and target agents are always fixed according to their characters such as their capabilities which have been predefined before the coordination. Therefore, if we can learn those partners who always closely coordinate, we are able to reorganize the team so that the average distance between those partners could be shortened.

According to this idea, we use a matrix called *source .target* to record messages' movements. Each element of $source.target[i][j]$ records the number of messages whose sender is agent a_i and receiver is agent a_j. In our learning process, when a message has been accepted (as shown in lines 11–15 and 21–24 in Algorithm 1), we are able to record its source (as a_i) and sink agent (as a_j) and $source.target[i][j]$ is accumulated by one. Based on the learned *source.target* matrix, it can be easily transformed into *connection* matrix:

$$connection\,[i]\,[j] = source.target\,[i]\,[j] + source.target\,[j]\,[i]\,. \tag{7}$$

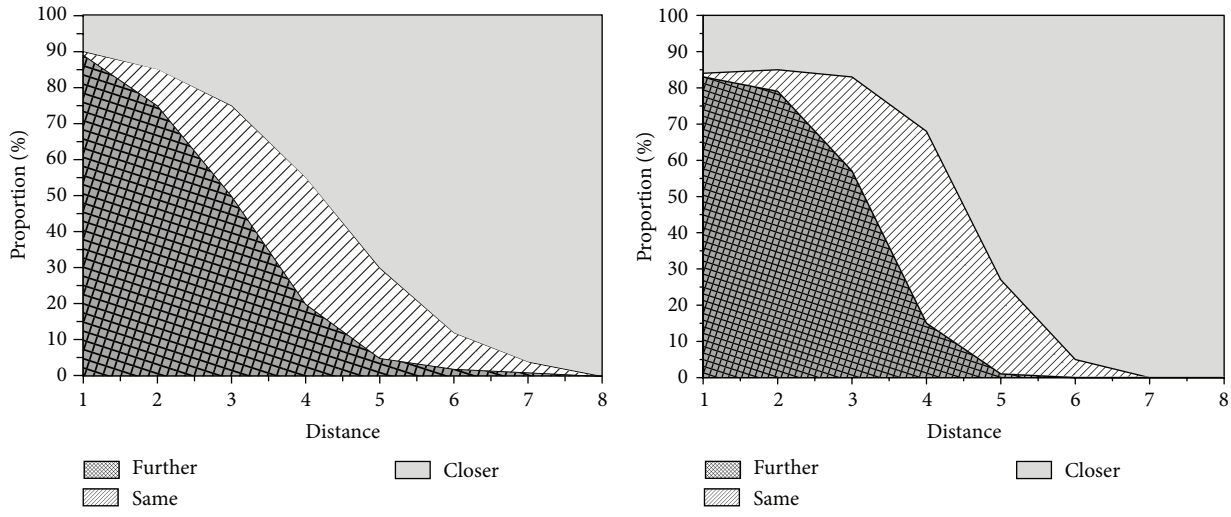
The *connection* matrix is symmetric and records the coordination between any pair of agents.

4.2. Subteams. Inspired by the clusters formed by closely cooperative individuals in human society, we build the *subteam* model to describe the similar group activities across the network. The *subteam* model, which is described in Figure 4 [11], is according to the mechanism of coordinative task planning. In the team coordination model we have built, the common goal is broken into subgoals $g_1, \ldots, g_i$, which can be executed by individual agents. Hence, agents can follow the planning mechanism to coordinate and achieve their subgoals.

Firstly, we predefined a number of plan templates in the library for agents to instantiate their plans. For example, when there is a fire in a building, the plan will be instantiated because it matches a template for disaster response. Each subgoal is addressed with a plan, $plan_i = \langle g_i, recipe_i, roles_i, d_i \rangle$, and thus the overall team plans, $Plans = \{plan_1, \ldots, plan_i\}$. g_i is the subgoal, and $recipe_i$ describes the way the subgoal will be achieved. $roles_i = \{r_1, \ldots, r_i\}$ are individual activities that must be performed to execute $recipe_i$, and d_i is the domain specific information pertinent to the plan.

Distributed plan creation is implemented by individual agents on behalf of the team, and we allow any member to commit the team to executing a plan when it detects that subgoal g_i is relevant. The subteams formation process commences when an individual agent detects all the appropriate preconditions that match a plan template in the library and subsequently instantiates a plan, $plan_i$. The $roles_i$ in $plan_i$ is embedded in the messages which are forwarded across the network until an agent finally accepts the role. Once accepted, the agent becomes a member of the subteam and makes a temporary commitment to perform the role toward the subgoal with the other subteam members. With the completion of the plan, the subteam will be dismissed so that new subteam for a new plan can be formed.

This algorithm can be described as an extended part of task allocation in team coordination. Algorithm 2 briefly



(a) In a scale-free network

(b) In a random network

Figure 2: The relative proportions of links that lead closer to the sink agent, keep the same distance, or move further from the sink agent, as the distance to it is varied [6].

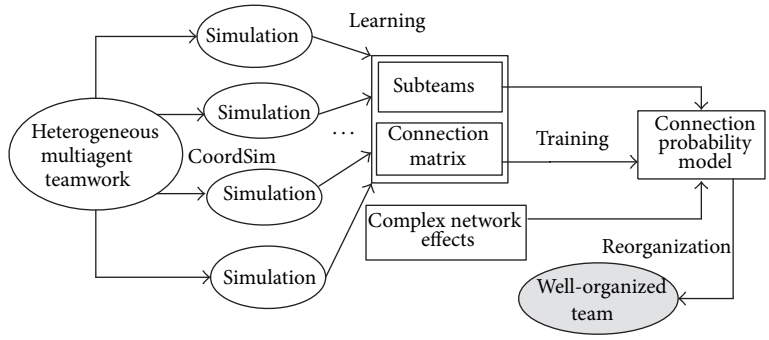


Figure 3: The team reorganization approach.

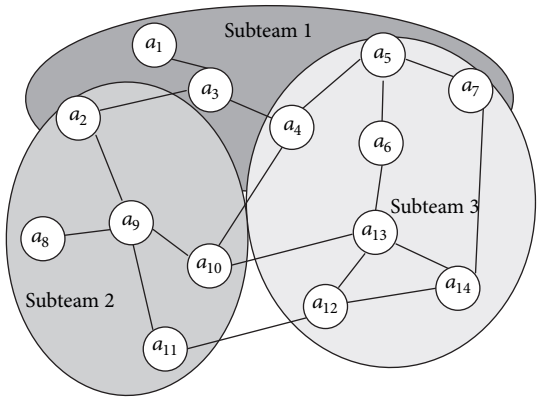


Figure 4: An example of the subteam model.

describes the formation of subteams. Although agents dynamically form and dismiss subteams, when agents form a subteam, they communicate frequently for their common interest. It should be learned and modeled in our adaptive algorithm so that the agents who were always in the same subteams should be connected closely to improve the team performance.

Based on the intrinsic character of the subteams, we consider it as an important factor for the team reorganization. In the coordination process, when agents use *JointIntentionMessage* to inform their neighbors to join a subteam, we record their subteam ID and mark it as $subID[i]$. Thus, we can learn the number of the subteams that agent i joined, written as $subteam[i]$. The overlapping subteam where agent i and agent j have joined together is written as $subteam[i, j]$. Therefore, the close relationship between i and j in the subteam model can be formalized as $sub[i][j]$:

$$sub\,[i]\,[j] = \frac{subteam\,[i, j] \times 2}{subteam\,[i] + subteam\,[j]}. \tag{8}$$

4.3. Integrated Reorganization Algorithm. In this section, we propose our adaptive reorganization algorithm. The key is to connect each pair of closely cooperated agents and promote

```
(1) for (m ∈ Messages) do
(2)   if (m is TaskMessages(α)) then
(3)     if (GetCap(α) > m.threshold) then
(4)       if (α ∉ OwnTasks) then
(5)         OwnTasks.append(α);
(6)         SendJointIntentionMessage(α.plan);
(7)         FormSubteams(α.plan);
(8)       end if
(9)     end if
(10)  end if
(11) end for
(12) for (α ∈ OwnTasks) do
(13)  if (α is complete) then
(14)    OwnTask.remove(α);
(15)    DismissSubteams(α.plan);
(16)  end if
(17) end for
```

Algorithm 2: Agents coordination (modified part).

```
(1) E = Φ;
(2) for (i = 1 to No_of_agents) do
(3)   for (j = 1 to Avg_degree/2) do
(4)     repeat
(5)       dest ← UniformRandom(Pr(i));
(6)     until Not(⟨i, dest⟩ ∈ E)
(7)     E.add(⟨i, dest⟩);
(8)   end for
(9) end for
(10) E.add(⟨i, dest⟩);
```

Algorithm 3: Integrated reorganization algorithm.

the helpful complex network attributes to design an integrated algorithm. In our algorithm, the closely coordinated relationships are defined according to the *connection matrix* and *subteams*. To build the integrated probability model, each probability of connecting agents a_i and a_j directly is written as $\Pr[i][j]$ ($\Pr[i][j] = \Pr[j][i]$). It is correlated with *connection matrix* and *subteams* according to the closely cooperative relationship, $d(j)$ according to the scale-free effect, and their distance distance(i, j) according to the small-world effect. Please note that we put a very small positive value ε as default in the probability model to guarantee that, although less likely, any agent is still able to directly connect with other agents. Specifically, we write $\Pr(i) = \{\Pr[i][j] \mid a_j \in A\}$ as the probability vector that a_i connects with any of the others.

The network reorganization algorithm is expressed as in Algorithm 3. In this algorithm, the network starts from empty link set (line 1). Each agent picks half of the predefined average degree of the network (line 3) and connects with them if they have not been connected (lines 5-6). The function *UniformRandom*$(\Pr(i))$ helps to pick agent a_i's neighbor based on its probability vector, which is the key to the

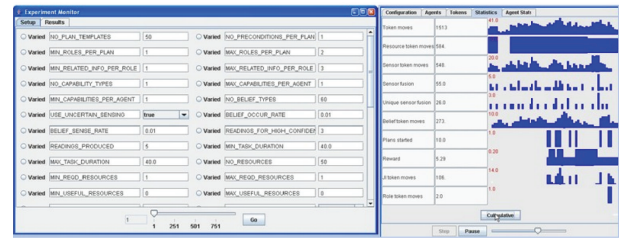


Figure 5: Screenshots of CoordSim simulator.

algorithm. In this paper, we briefly consider four key factors to build the probability $\Pr(i, j)$:

(i) $connection[i][j] / \sum_{a_k \in A} connection[i][k]$ models the preference of agent i directly connecting j according to the connection matrix;

(ii) $sub[i][j] / \sum_{a_k \in A} sub[i][k]$ models the preference of agent i directly connecting j according to the *subteam* model;

(iii) $d(j) / \sum_{a_k \in A} d(k)$ models j's degree distribution to infer whether i should connect to j to promote scale-free effect. The higher the degrees of j are, the more likely the i will connect to j;

(iv) $distance(i, j) / \sum_{a_k \in A} distance(i, k)$ models the small word effects. The further the distance between i and j is, the more likely they will be directly connected.

If agent a_i starts a new connection, the probability of connecting with a_j is defined as

$$\begin{aligned}\Pr(i, j) &= \beta \times \left(\frac{connection[i][j]}{\sum_{a_k \in A} connection[i][k]} + \frac{sub[i][j]}{\sum_{a_k \in A} sub[i][k]} \right) \\ &\quad + \gamma \times \frac{d(j)}{\sum_{a_k \in A} d(k)} + \lambda \times \frac{distance(i, j)}{\sum_{a_k \in A} distance(i, k)},\end{aligned} \tag{9}$$

where normalization should be applied. $\{\beta, \gamma, \lambda\}$ are the important factors and $\beta + \gamma + \lambda = 1$. Please note that if $\{\beta = 1, \gamma = 0, \lambda = 0\}$, we only take the character of closely coordinative relationships into consideration; if $\{\beta = 0, \gamma = 1, \lambda = 0, \}$, we will set up a standard scale-free network; and if $\{\beta = 0, \gamma = 0, \lambda = 1\}$, we will set up a standard small-world network [12].

5. Simulations and Results

To simulate the real team coordination, we use our abstract simulator called CoordSim [15]. This simulator is capable of simulating the major aspects of coordination, including task assignment and resource allocation. CoordSim abstracts the environment by simulating only its effects on the team. According to the team coordination process in Section 2.3, reward is simulated as being received by the team when a task is allocated. CoordSim allows a large number of parameters

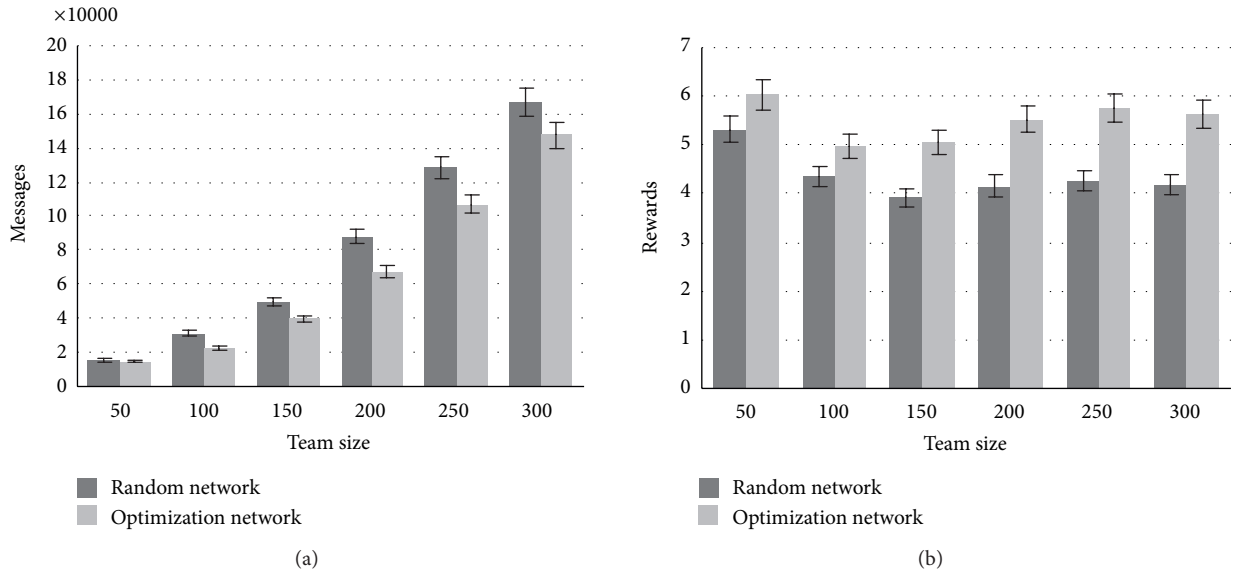


Figure 6: The reorganization algorithm is scalable on different sizes of teams.

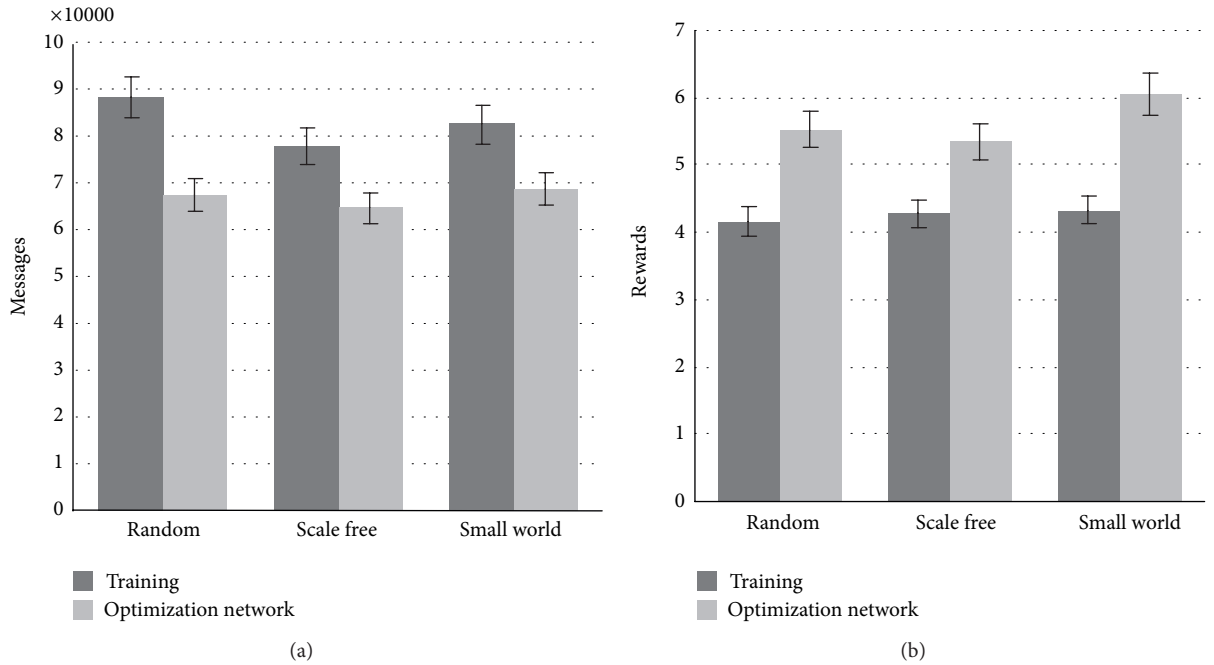


Figure 7: The reorganization algorithm improves the team performances when the teams originally organized as different network structures.

to be varied and also allows statistics to be recorded, such as the number of rewards and message movements, which is important to our approach. Two interfaces of this simulator are shown in Figure 5. There are more than 20 parameters that can be varied, covering the major aspects of the large heterogeneous coordination.

If not otherwise stated, the experiments are configured as follows. There are 100 agents deployed in a 500 × 500 environment to perform 100 tasks with 100 resources. Each task requires at least one resource. In the default setup, the heterogeneous team has five different types of capabilities. Task and resource messages are allowed to move unless accepted. "Reward" is the sum of the rewards received by each agent. "Messages" is the number of times that agents communicated for coordination. The objective of the team coordination is to gain the best tradeoff between maximizing team rewards and minimizing messages. To build intelligent coordination between agents, we used the literature [13] as the coordination schema in the simulations. Simulation will last for 500 time steps. All of the experimental results are based

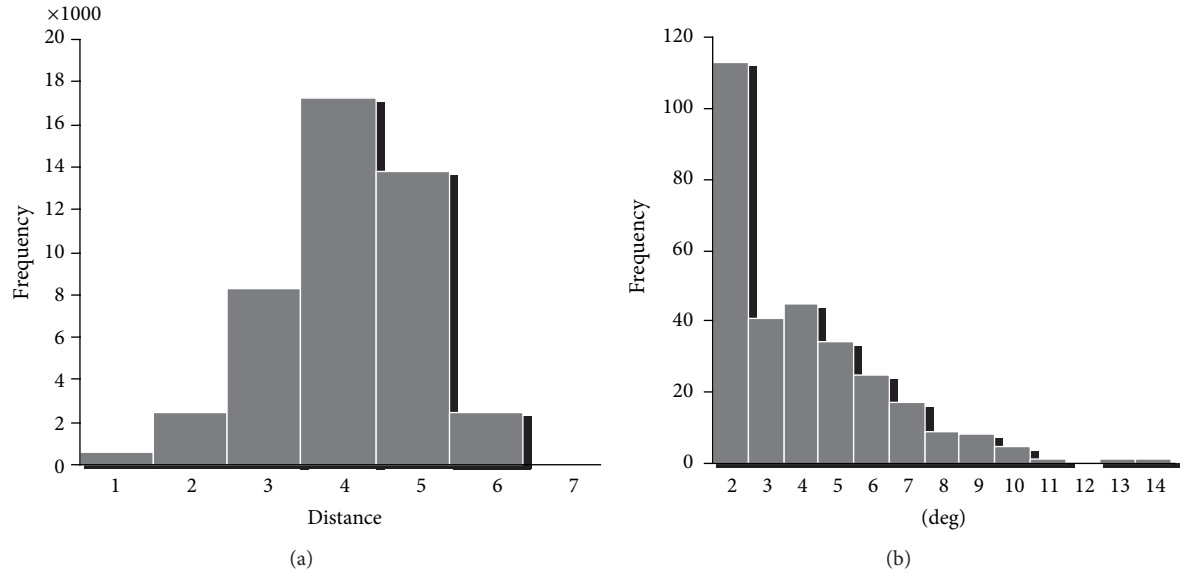


Figure 8: After reorganization, the team has small-world and scale-free effects.

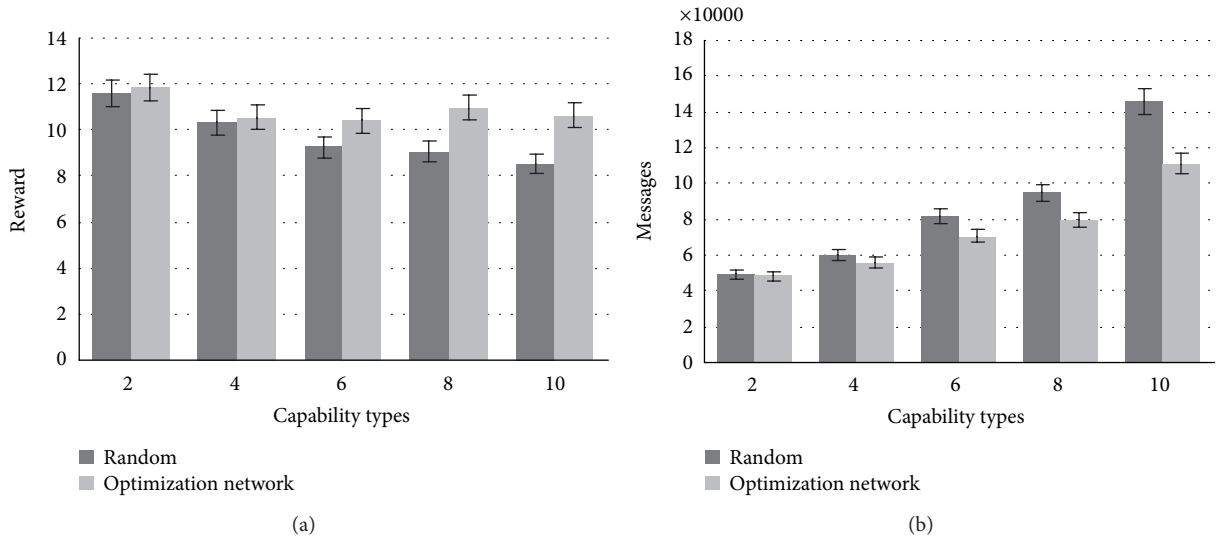


Figure 9: The reorganization algorithm improves the heterogeneous teams more significantly.

on 100 runs. To build the *connection matrix* and *subteam*, we sampled the team with 100 runs as well. In each run, all the agents and environmental settings keep the same, but their organization networks are different.

We evaluated the efficiency of our team reorganization algorithm in five experiments. In the first experiment, we first organized the team as random networks and set $\{\beta = 1, \gamma = 0, \lambda = 0\}$ that the team should be reorganized according to the *connection matrix* and *subteam* only. We varied the team size from 50 to 300, and to be fairly compared, tasks and resources in large teams were more to keep the same tasks per agent and resources per agent. The experimental results in Figure 6 show that, no matter what the team sizes are, the reorganized network outperforms the random team organization with higher rewards and less communication costs.

In the second experiment shown in Figure 7, we organized the original network with three different structures: random, small world, and scale free. The team size is 100 and we set $\{\beta = 1, \gamma = 0, \lambda = 0\}$ that the team will be reorganized according to the connection matrix and subteam model only. As we expected, sine we have taken the team's intrinsic closely cooperative relationship between heterogeneous agents into consideration, the team performance is greatly improved no matter what the original network is.

In Figure 8, we investigate whether the reorganized networks have the complex network attributes. The reorganized

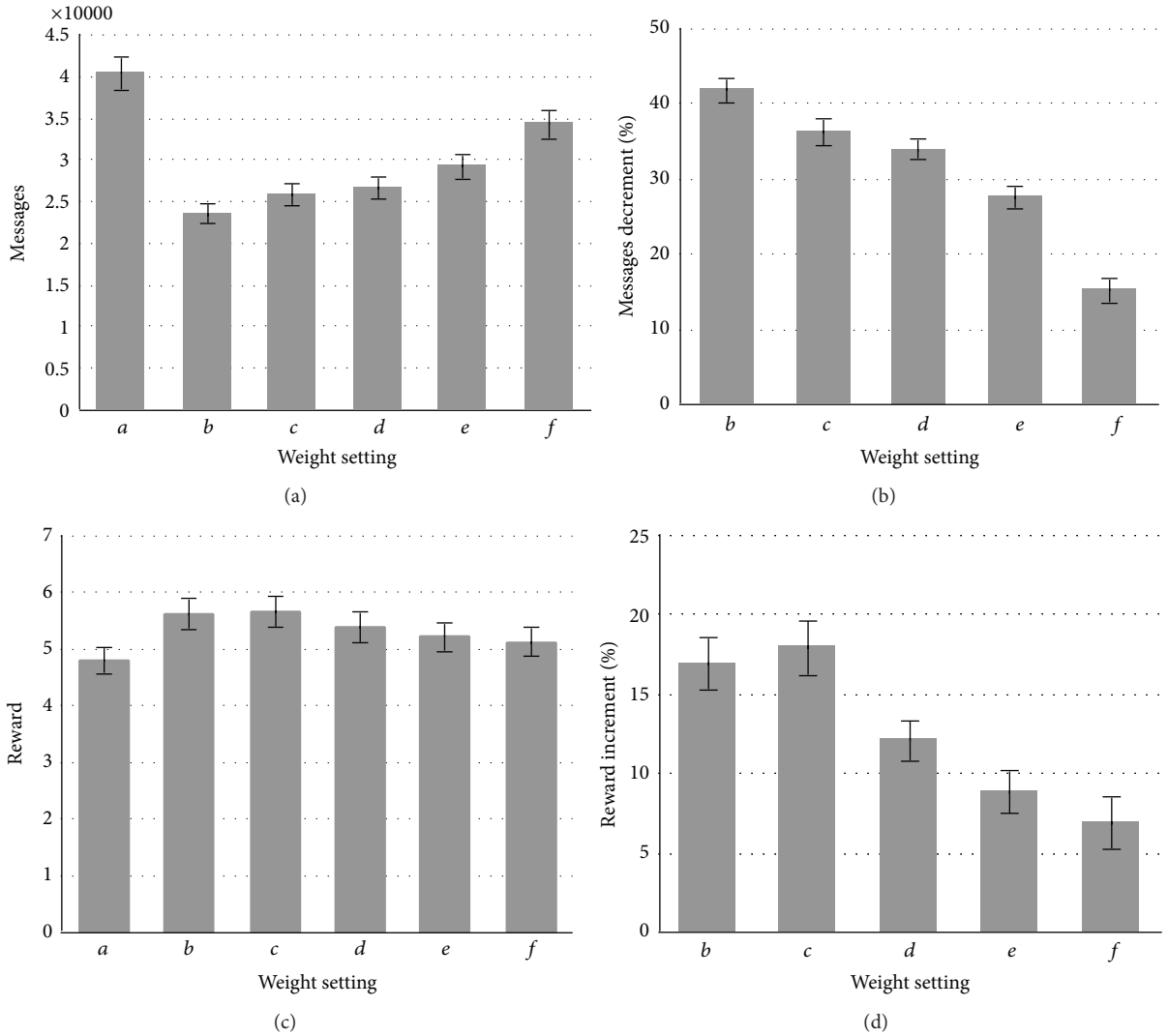


Figure 10: The integrated algorithm improves the team performance when β is higher than the others.

network is learned from a random network and the team size is 100. We set $\{\beta = 1, \gamma = 0, \lambda = 0\}$ as well. The distribution of distance and degree is shown in Figure 8. We can see that the reorganized network has the small-world and scale-free effects.

In the fourth experiment, we verify our design in different heterogeneous teams. The team size is 100 and the original network is random network. We set $\{\beta = 1, \gamma = 0, \lambda = 0\}$ as before. Agents' capability types are set from 2 to 10 to make the team more and more heterogeneous. Note that when team becomes more heterogeneous, less agents are capable of accepting a given task message. It will make the coordination very hard. As the experiment results in Figure 9 show, the reward gains keep decreasing in the original random networks; however, it is not the case in our reorganized networks. We hypothesize that when team becomes more heterogeneous, the closely cooperative relationship is more prominent to be caught, which makes our design more efficient.

In the last experiment, we set up the original network as random networks and team size is 100. We have six settings: $a = \{\beta = 0.0, \gamma = 0.0, \lambda = 0.0\}$, $b = \{\beta = 1.0, \gamma = 0.0, \lambda = 0.0\}$, $c = \{\beta = 0.7, \gamma = 0.02, \lambda = 0.28\}$, $d = \{\beta = 0.5, \gamma = 0.05, \lambda = 0.45\}$, $e = \{\beta = 0.3, \gamma = 0.05, \lambda = 0.65\}$, and $f = \{\beta = 0, \gamma = 0.05, \lambda = 0.95\}$ so that the team organization is set according to all the three factors, but with higher portions to small-world and scale-free effects. Experimental results in Figure 10 show that, comparing with a which is not reorganized, by integrating all three factors, the team performance with reorganization is improved. However, to gain the best performance, $c = \{\beta = 0.7, \gamma = 0.02, \lambda = 0.28\}$ works the best. When $f = \{\beta = 0, \gamma = 0.05, \lambda = 0.95\}$, the performance is worse with the lack of closely cooperative relationship in organization. We explain that little portions

of complex network effects help the team, but discovering the closely cooperative relationship contributes the most.

6. Related Work

Many researchers have demonstrated that the organizational design in multiagent systems has a significant effect on its performances [16], and the properties of small-world network can enhance network's signal propagation speed, computational power, and synchronization [17]. Glinton et al. [18] have found that the team is of high performance and rapid convergence when the scale-free organization was formed, and limited links per agent in the complex network improve team performance.

A range of organizational strategies have been proposed to improve multiagent team. In literature [19], it presented a distributed scenario for team formation in multiagent systems and concluded that the direct interconnections among agents were determined by the agent interaction topology. Kota et al. [20] provided a self-organization method which enabled agents to modify the structural relations to achieve a better allocation of tasks. Keogh and Sonenberg [21] designed a flexible, coordinated organization-based agent system in which agents can adjust their own attitudes to fit in others in a changing situation by having the access to organizational information that they can change. A composite self-organization mechanism in a multiagent network is proposed in [22]. It enables agents to dynamically adapt team organization by using a trust model and the former task allocation to assist agents to decide whom they should connect with. Terauchi et al. [23] proposed an agent organization management system and provided context based organizational information for problem solving. It is only concerned with improving system scalability and flexibility but neglected the coordination efficiency.

7. Conclusion and Future Works

In this paper, we have made an initial effort on finding how to adjust team organization to expedite team performance. We have proved that team organization is a key factor to the team performance. Based on scalable team coordination schema, we designed an adaptive team organization algorithm by incorporating the nature of agents' cooperation relationship and the attributes of complex network. Our experiments have been proven to the validity of our design.

While this work represents an important step in this regard, much work remains to be done. First of all, we have only designed an offline learning algorithm to adjust the team, while online adjustment may be available. Secondly, our approach is based on the organization of associated network which is based on a P2P coordination infrastructure, while in many application domains, the coordination is based on broadcast.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This research has been sponsored by NSFC 61370151, 60905042, and 61202211, National Science and Technology Support Program of China 2012BAI22B05, and Central University Basic Research Funds Foundation ZYGX2011X013.

References

[1] J. Han, C. Wang, and G. Yi, "Cooperative control of UAV based on multi-agent system," in *Proceedings of the 8th IEEE Conference on Industrial Electronics and Applications*, 2013.

[2] T. Yang, M. Wang, and M. Fang, "The military strategy threat of cognitive system based on agent," in *Proceedings of the International Conference of Modern Computer Science and Applications Advances in Intelligent Systems and Computing*, vol. 191, 2013.

[3] M. Mala, "RCAIMS: a reactive multi-agent system based incident/emergency management system," *International Journal of Computational Intelligence Studies*, vol. 2, no. 3-4, pp. 300–313, 2013.

[4] V. Gorodetsky, O. Karsaev, V. Samoylov, and S. Serebryakov, "P2P agent platform: implementation and testing," in *Agents and Peer-to-Peer Computing*, vol. 5319 of *Lecture Notes in Computer Science*, pp. 41–54, Springer, Berlin, Germany, 2010.

[5] M. E. Gaston and M. DesJardins, "Agent-organized networks for dynamic team formation," in *Proceedings of the 4th International Conference on Autonomous Agents and Multi agent Systems (AAMAS '05)*, pp. 375–382, July 2005.

[6] P. Scerri and K. Sycara, "Social networks for effective teams," *Cooperative Networks: Control and Optimization*, 2008.

[7] Y. C. Jiang and J. C. Jiang, "Understanding social networks from a multi-agent coordination perspective," *IEEE Transactions on Parallel and Distributed Systems*, 2013.

[8] J. Pitt, D. Ramirez-Cano, M. Draief, and A. Artikis, "Interleaving multi-agent systems and social networks for organized adaptation," *Computational and Mathematical Organization Theory*, vol. 17, no. 4, pp. 344–378, 2011.

[9] Q. Han, T. Arentze, H. Timmermans, D. Janssens, and G. Wets, "The effects of social networks on choice set dynamics: Results of numerical simulations using an agent-based approach," *Transportation Research A: Policy and Practice*, vol. 45, no. 4, pp. 310–322, 2011.

[10] Y. Jiang, Y. Zhou, and W. Wang, "Task allocation for undependable multiagent systems in social networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 8, pp. 1671–1681, 2013.

[11] E. Liao, P. Scerri, and K. Sycara, "A framework for very large teams," in *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems Workshop on Coalitions and Teams*, 2004.

[12] R. Albert, H. Jeong, and A. L. Barabási, "Diameter of the world-wide web," *Nature*, vol. 401, no. 6749, pp. 130–131, 1999.

[13] Y. Xu, P. Scerri, B. Yu, S. Okamoto, M. Lewis, and K. Sycara, "An integrated token-based algorithm for scalable coordination," in *Proceedings of the 4th International Conference on Autonomous Agents and Multi agent Systems*, pp. 543–550, July 2005.

[14] L. Lovasz, "Random walks on graphs: a survey," in *Combinatorics, Bolyai Mathematical Society*, 1993.

[15] Y. Xu, P. Scerri, K. Sycara, and M. Lewis, "Comparing market and token-based coordination," in *Proceedings of the 5th*

International Conference on Autonomous Agents and Multiagent Systems, pp. 1113–1115, May 2006.

[16] S. van Segbroeck, S. de Jong, A. Nowé, F. C. Santos, and T. Lenaerts, "Learning to coordinate in complex networks," *Adaptive Behavior*, vol. 18, no. 5, pp. 416–427, 2010.

[17] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[18] R. Glinton, K. Sycara, and P. Scerri, "Agent organized networks redux," in *Proceedings of the AAAI Conference on Artificial Intelligence and the 20th Innovative Applications of Artificial Intelligence Conference*, pp. 83–88, July 2008.

[19] L. Coviello and M. Franceschetti, "Distributed team formation in multi-agent systems: stability and approximation," in *Proceedings of the 51st IEEE Conference on Decision and Control*, 2012.

[20] R. Kota, N. Gibbins, and N. R. Jennings, "Self-organising agent organisations," in *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems*, 2009.

[21] K. Keogh and L. Sonenberg, "Adaptive coordination in distributed and dynamic agent organizations," in *Coordination, Organizations, Institutions, and Norms in Agent System VII*, vol. 7254 of *Lecture Notes in Computer Science*, pp. 38–57, Springer, Berlin, Germany, 2012.

[22] D. Ye, M. Zhang, and D. Sutanto, "Self-organization in an agent network: a mechanism and a potential application," *Decision Support Systems*, vol. 53, no. 3, pp. 406–417, 2012.

[23] A. Terauchi, O. Akashi, M. Maruyama et al., "ARTISTE: agent organization management system for multi-agent systems," in *Multi-Agent Systems for Society*, vol. 4078 of *Lecture Notes in Computer Science*, pp. 207–221, Springer, Berlin, Germany, 2009.