Self-organisation in an Agent Network via Multiagent Q-Learning

Dayong Ye[1], Minjie Zhang[1], Quan Bai[2], and Takayuki Ito[3]

[1] University of Wollongong, Wollongong, NSW 2522 Australia
[2] CSIRO ICT Centre, Hobart, TAS 7001 Australia
[3] Nagoya Institute of Technology, Nagoya 466-8555 Japan
{dy721,minjie}@uow.edu.au, Quan.Bai@csiro.au, ito@nitech.ac.jp

Abstract. In this paper, a decentralised self-organisation mechanism in an agent network is proposed. The aim of this mechanism is to achieve efficient task allocation in the agent network via dynamically altering the structural relations among agents, i.e. changing the underlying network structure. The mechanism enables agents in the network to reason with whom to adapt relations and to learn how to adapt relations by using only local information. The local information is accumulated from agents' historical interactions with others. The proposed mechanism is evaluated through a comparison with a centralised allocation method and the *K-Adapt* method. Experimental results demonstrate the decent performance of the proposed mechanism in terms of several evaluation criteria.

1 Background

To cope with complex tasks, agents are usually organised in a network where an agent interacts only with its immediate neighbours in the network. Self-organisation, defined by Serugendo et al. [1] as "*the mechanism or the process enabling the system to change its organisation without explicit external command during its execution time*", can be used in agent networks to improve the cooperative behaviours of agents. Mathieu et al. [2] provided three principles for self-organisation agent networks design, including (1) creation of new specific relations between agents in order to remove the middle-agents, (2) exchange of skills between agents to increase autonomy, and (3) creation of new agents to reduce overloading. In this paper, our contribution focuses on the first principle provided by Mathieu et al., i.e. modification of existing relations between agents to achieve a better allocation of tasks in distributed environments.

Currently, research on self-organisation mechanisms in multiagent and agent-based complex systems has produced results of significance. Horling et al. [3] proposed an organisation structure adaptation method, which employed a central blackboard, by using self-diagnosis. Their method involves a diagnostic subsystem for detecting faults in the organisation. Then, against these faults, some fixed pre-designed reorganisation steps are launched. Hoogendoorn [4] presented an approach based on max flow networks to dynamically adapt organisational

B.-H. Kang and D. Richards (Eds.): PKAW 2010, LNAI 6232, pp. 14–26, 2010.

models to environmental fluctuation. His approach assumed two special nodes in the network, namely the *source node* with indegree of 0 and the *sink node* with outdegree of 0. These two special nodes make the approach centralised, since if these two nodes are out of order the organisation adaptation process might be impacted. Hence, the approaches proposed by both Horling et al. [3] and Hoogendoorn [4] are centralised in nature and have the potential of single point failure. Kamboj and Decker [5] developed a self-organisation method based on organisational self-design, which achieved self-organisation by dynamically generating and merging agents in response to the changing requirements. Nevertheless, the internal framework of agents might not be changed on all occasions because of physical and accessibility limitations (e.g. a remote central server, which is modeled as an agent, cannot easily be replicated).

Self-organisation methods focusing on network structural adaptation (namely modifying relations among agents), which are used to improve team formation or task allocation, have also been researched. Gaston and desJardins [6] developed two network structural adaptation strategies for dynamic team formation, i.e. structure-based approach and performance-based approach. The two strategies are suitable in different situations. Glinton et al. [7] empirically analysed the drawback of the structure-based strategy proposed by Gaston and desJardins [6], and then designed a new network adaptation strategy which limits the maximum number of links an agent can have. Abdallah and Lesser [8] did further research in self-organisation of agent networks and creatively used reinforcement learning to adapt the network structure by allowing agents to not only adapt the underlying network structure during the learning process but also use information from learning to guide the adaptation process. The common limitation of these works is that they assumed that only one type of relations exists in the network and the number of neighbours possessed by an agent has no effect on its local load. These assumptions are impractical in some cases where multiple relations exist among agents in a network and agents have to expend resources to manage their relations with other agents. To overcome this common limitation, Kota et al. [9] [10] devised a network structural adaptation mechanism through meta-reasoning. Their mechanism primarily supposed that multiple relations exist in the network and considered the load of agents to manage their relations. Nonetheless, their mechanism is somewhat biased from the agent which initialises the relation adaptation process towards the agent which is requested to accessorily adapt relation, because the initiative agent evaluates most attributes regarding relation adaptation. This bias might cause the initiative agents a little subjective when making decisions.

Against this background, in this paper, we propose a self-organisation mechanism, called *Learn-Adapt*. This mechanism adopts a token based approach, and can achieve decentralised structural adaptations via multiagent Q-learning. Xu et al. [11] have shown that token-based mechanisms can collect as much information as broadcast mechanisms while using much less bandwidth. Compared with current approaches, our mechanism utilises only local information to adapt the network structure and considers multiple relations in the network,

communication efficiency of different relations, the management cost effect on agents which is brought by their neighbours, and the potential benefit after changing relations. According to our mechanism, when an agent intends to change their relation with another agent, the two agents independently evaluate their rewards about changing relations through learning the Q-value of each available action and then the two agents jointly make a reasonable and optimal decision about changing relations. In contrast to the *K-Adapt* mechanism, proposed by Kota et al. [10], our *Learn-Adapt* mechanism is unbiased for both agents which jointly adapt their relation, and we empirically demonstrate that the performance of our mechanism, *Learn-Adapt*, is better than *K-Adapt* in different situations. The rest of the paper is organised as follows. Section 2 introduces our agent network model for task allocation. Section 3 illustrates the network performance evaluation approach and proposes a decentralised network structural adaptation mechanism. Experimental results and analysis are presented in Section 4. The paper is concluded in Section 5.

2 The Agent Network Model

The aim of the agent network is to assign tasks to agents such that the communication cost among agents is minimised and the benefit obtained by completing tasks is maximised. In our model, an agent network comprises a set of collaborative agents, i.e. $A = \{a_1, ..., a_n\}$, situated in a distributed task allocation environment. The task environment presents a continuous dynamic stream of tasks that have to be performed. Each task, Θ, is composed of a set of subtasks, i.e. $\Theta = \{\theta_1, ..., \theta_m\}$. Each subtask, $\theta_i \in \Theta$, requires a particular resource and a specific amount of computation capacity to fulfill. In addition, each subtask has a relevant benefit paid to the agent which successfully completes the subtask. A subtask θ_i is modeled as a token Δ_i which can be passed in the network to find a suitable agent to complete. Each token consists of not only the information about resource and computation requirement of the corresponding subtask, but also the token traveling path which is composed of those agents that the token has passed.

In the agent network, instead of a single type of neighbours, there are three types of neighbours, namely *peer*, *subordinate* and *superior* neighbours, which are constituted by two relations, i.e. *peer-to-peer* and *subordinate-superior* relations. The formal definitions of these two relations are given below.

Definition 1. (*Peer-to-Peer*). A *peer-to-peer relation*, denoted as "$\sim$" ($\sim \subseteq A \times A$), is a *Compatible Relation*, which is reflexive and symmetric, such that $\forall a_i \in A : a_i \sim a_i$ and $\forall a_i, a_j \in A : a_i \sim a_j \Rightarrow a_j \sim a_i$.

Definition 2. (*Subordinate-Superior*). A *subordinate-superior relation*, written as "$\prec$" ($\prec \subseteq A \times A$), is a *Strict Partial Order Relation*, which is irreflexive, asymmetric and transitive, such that $\forall a_i \in A : \neg(a_i \prec a_i)$, $\forall a_i, a_j \in A : a_i \prec a_j \Rightarrow \neg(a_j \prec a_i)$ and $\forall a_i, a_j, a_k \in A : a_i \prec a_j \wedge a_j \prec a_k \Rightarrow a_i \prec a_k$.

According to the above definitions, there are three neighbour types, which are *peer*, *subordinate* and *superior* neighbours. For convenience, we stipulate that relation $\succ$ is the reverse relation of $\prec$, namely $a_i \prec a_j \Leftrightarrow a_j \succ a_i$. In this paper, it is assumed that there is at most one relation between two agents in the network. Figure 1 displays an example agent network. The contents in parenthesis denote the resources supplied by an agent.

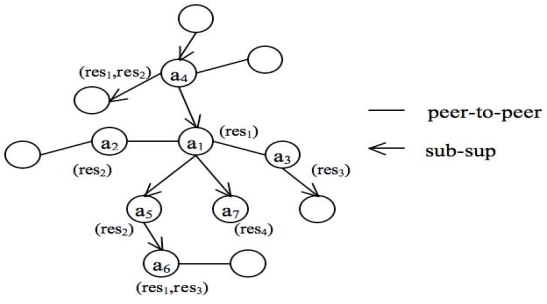


Fig. 1. An Example Agent Network at Time t

Each agent is of the form $a_i = \langle Res_i, Comp_i \rangle$, where Res_i is the set of resources possessed by a_i and $Comp_i$ is the computation capacity of a_i for completing tasks. Moreover, the knowledge of each agent is modeled as a tuple $\langle Neig_i(t), Act_i(t), Tokens_i(t) \rangle$. The first element, $Neig_i(t)$, is the neighbours set of agent a_i at time t. $Neig_i(t)$ can be further divided into three subsets, $Neig_i^{\sim}(t)$, $Neig_i^{\prec}(t)$ and $Neig_i^{\succ}(t)$. $Neig_i^{\sim}(t)$ contains the *peer*s of a_i, $Neig_i^{\prec}(t)$ is consisted of the direct *superior*s of a_i, and $Neig_i^{\succ}(t)$ comprises of the direct subordinates of a_i.

The second element in agent knowledge tuple, $Act_i(t)$, is the available action set of agent a_i at time t. Before description of the available action set, we first introduce the definition of action.

Definition 3. (*Action*) An *action* is defined as a decision made by an agent to change relation with another agent.

There are seven different atomic actions defined in our model, which are $form\_\sim$, $form\_\prec$, $form\_\succ$, $dissolve\_\sim$, $dissolve\_\prec$, $dissolve\_\succ$ and *no_action*. For example, if agent a_i performs action $form\_\prec$ with agent a_j, a_i will become a *subordinate* of a_j. Obviously, actions $form\_\prec$ and $form\_\succ$ are the reverse action of each other, namely that if a_i performs $form\_\prec$ with a_j, a_j has to take $form\_\succ$ with a_i. The atomic actions can be combined together. The meanings of combination actions can be easily deduced from the meanings of atomic actions. For example, the combination action, $dissolve\_\prec + form\_\sim$, which is taken by a_i with a_j, implies that a_i first dissolves a_j from a_i's *superior* and forms a *peer* relation with a_j. It should be noted that an agent at different time steps might possess different available actions.

The possible choices of actions available to agents in different situations are illustrated as follows.

1. There is no relation between agents a_i and a_j. The possible choices of actions include $form\_\sim$, $form\_\prec$, $form\_\succ$ and *no_action*.

2. a_i is a *peer* of a_j, i.e. $a_i \sim a_j$. The possible actions involve $dissolve\_\sim$, $dissolve\_\sim + form\_\prec$, $dissolve\_\sim + form\_\succ$ and *no_action*.

3. a_i is a *subordinate* of a_j, i.e. $a_i \prec a_j$. The possible actions include $dissolve\_\prec$, $dissolve\_\prec + form\_\sim$, $dissolve\_\prec + form\_\succ$ and *no_action*. These actions are based on a_i's perspective, while, on a_j's view, a_j needs to reverse these actions.

4. a_i is a *superior* of a_j, i.e. $a_j \prec a_i$. This situation is the reverse condition of $a_i \prec a_j$.

Definition 4. (*Action Set*). An *action set*, $Act_i(t)$, is defined as a set of available actions for the agent a_i at time t, which includes some atomic actions or combination actions.

The last element in the agent knowledge tuple, $Tokens_i(t)$, stores not only the tokens agent a_i currently holds at time t but also all the previous tokens incoming and outgoing through a_i. In this paper, we assume that each agent has an infinite amount of memory to store tokens.

Furthermore, an agent possesses information about the resources it provides, the resources its *peer*s could provide, and the resources all of its *subordinate*s and its direct *superior* could provide, although the agent might have no idea exactly which *subordinate* owns which resource.

During the allocation of a subtask θ, an agent a_i always tries to execute the subtask by itself if it has adequate resources and computation capacity. Otherwise, a_i will generate a token for the subtask and pass the token to one of its *subordinate*s which contains the expected resource. Since a_i does not know which *subordinate* has which *resource*, the token might be passed several steps in the agent network forming a delegation chain. If a_i finds no suitable subordinate (no subordinate contains the expected resource), it will try to pass the token to its *peer*s. In the case that no peer is capable of the subtask, a_i will pass the token back to one of its *superior*s which will attempt to find some other subordinates or peers for delegation.

Apparently, the structure of the agent network will influence the task allocation process. In the next section, we will describe the self-organisation mechanism to adapt the structure of the agent network, involving an evaluation method to measure the profit of the network.

3 Decentralised Self-organisation Mechanism

Before devising a decentralised self-organisation mechanism, it is necessary to introduce an evaluation method to estimate the profit of the agent network. Towards this goal, we illustrate the concept of evaluation criteria, which includes cost, benefit, profit and reward of an agent and, further, the agent network.

3.1 Network Performance Evaluation

The cost, benefit and profit of the network are calculated after a predefined number of tasks are allocated, each of which contains several subtasks. The cost of the agent network, $Cost_{NET}$, consists of four attributes, i.e. communication cost, computation cost consumed by agents to complete assigned subtasks, management cost for maintaining subtasks and management cost for keeping neighbour relations with other agents. Due to the page limitation, the detailed calculation of each attribute of $Cost_{Net}$ cannot be presented.

The benefit of the network, $Benefit_{NET}$, is simply the sum of benefits obtained by all the agents in the network. The benefit of each agent depends on how many subtasks are completed by that agent. As depicted in Section 2, each task Θ contains several subtasks, $\theta_1, \theta_2, ...$, represented as tokens Δ_1, Δ_2.... When a subtask θ_i is successfully completed, the agent which executes this subtask can obtain the relevant benefit.

Finally, the profit of the entire network, $Profit_{NET}$, is:

$$Profit_{NET} = Benefit_{NET} - Cost_{NET} \tag{1}$$

3.2 Self-organisation Mechanism Design

The aim of our self-organisation mechanism is to improve the profit of the agent network during task allocation processes via changing the network structure, i.e. changing the relations among agents. Our mechanism is based on the historical information of individual agents. Specifically, agents use the information about the former task allocation processes to evaluate their relations with other agents. We formulate our self-organisation mechanism by using a multiagent Q-learning approach. The reason for choosing the Q-learning approach is that it provides a simple and suitable methodology for representing our mechanism in terms of actions and rewards. Before describing our self-organisation mechanism, we first consider a simple scenario with two agents, a_i and a_j, and three available actions for each agent. The reward matrix of the two agents is displayed in Table 1.

Each cell $(r_i^{x,y}, r_j^{x,y})$ in Table 1 represents the reward received by the row agent (a_i) and the column agent (a_j), respectively, if the row agent a_i plays action x and the column agent a_j plays action y.

The reward of each agent, r_i, is based on how much load could be reduced on agent a_i and how much load could be decreased on the intermediate agents,

Table 1. Reward Matrix of a_i and a_j

a_i \ a_j	$form\_\prec$	$form\_\sim$	$form\_\succ$
$form\_\succ$	$r_i^{1,1}, r_j^{1,1}$	$r_i^{1,2}, r_j^{1,2}$	$r_i^{1,3}, r_j^{1,3}$
$form\_\sim$	$r_i^{2,1}, r_j^{2,1}$	$r_i^{2,2}, r_j^{2,2}$	$r_i^{2,3}, r_j^{2,3}$
$form\_\prec$	$r_i^{3,1}, r_j^{3,1}$	$r_i^{3,2}, r_j^{3,2}$	$r_i^{3,3}, r_j^{3,3}$

and how much potential benefit might be obtained by agent a_i in the future. Here, an intermediate agent is an agent which resides on a token path, written as $\Delta.path$. For example, agent a_i has no relation with agent a_j, but a_i received many subtasks from a_j during former task allocation processes. a_i, then, makes the decision with regard to forming a relation with a_j. If a_i would like to form the *subordinate-superior* relation with a_j, i.e. performing the action $form\_\prec$ (for a_j, performing the reverse action $form\_\succ$), the management cost on both a_i and a_j will rise because both a_i and a_j have to maintain a new neighbour. Nevertheless, those agents, which are in the $\Delta.path$, could save communication cost (which are considered as a_i's reward when a_i makes decisions), since they do not need to pass tokens between agents a_i and a_j any more. Here, Δ refers to the tokens that are held by a_i and sent by a_j. For a_j, it could save management cost for maintaining subtasks, as a_j can directly pass tokens to a_i without waiting for intermediate agents to pass tokens and, hence, a_j's subtasks could be allocated in less time steps. The potential benefit which would be obtained by a_i is evaluated on the basis of the benefit a_i gained for completing the subtasks assigned by a_j, while the potential benefit of a_j is calculated in an analytical way. We suppose that the action $form\_\prec$, with a_i as the *subordinate* and a_j as the *superior*, can make a_i potentially receive more subtasks from a_j and then get more benefits. *Algorithm 1* demonstrates our self-organisation mechanism in pseudocode form.

Algorithm 1. Reorg. Mechanism according to a_i

```
1  Candidates_i ← a_i selects agents in the network;
2  for each a_j ∈ Candidates do
3      Act_i ← available_actions(a_i, a_j);
4      Act_j ← available_actions(a_i, a_j);
5      for each x ∈ Act_i, y ∈ Act_j do
6          Initialise Q_ix and Q_jy arbitrarily;
7          for k = 0 to a predefined integer do
8              calculate π_ix(k) and π_jy(k);
9              Q_ix(k+1) = Q_ix(k)+
                 π_ix(k)α(Σ_y r_i^{x,y} π_jy(k) − Q_ix(k));
10             Q_jy(k+1) = Q_jy(k)+
                 π_jy(k)α(Σ_x r_j^{x,y} π_ix(k) − Q_ix(k));
11         end for
12     end for
13     ⟨x_opti, y_opti⟩ ← argMax_{match(x,y)}(Q_ix + Q_jy);
14     a_i, a_j take actions x_opti and y_opti, respectively;
15     end if
16 end for
```

The first component (Line 1) refers to the reasoning aspect, which is displayed in *Algorithm 2*, about selecting agents to initiate the self-organisation process. After selection, both agents, a_i and a_j, estimate which actions are available at the current state (Lines 3 and 4) as described in Section 2. Then, a_i and a_j learn the Q-value of each available action, separately (Lines 5-11). In Line 6,

the Q-value of each action is initialised arbitrarily. In Line 8, π_{ix} indicates the probability regarding agent a_i taking the action x. To calculate π_{ix}, we employ the ϵ-greedy exploration method devised by Gomes and Kowalczyk [12] shown in Equation 2, where $0 < \epsilon < 1$ is a small positive number and n is the number of available actions possessed by agent a_i.

$$\pi_{ix} = \begin{cases} (1-\epsilon) + (\epsilon/n), \text{if } Q_{ix} \text{ is the highest} \\ \epsilon/n \text{ , otherwise} \end{cases} \tag{2}$$

Furthermore, in Line 9, Q_{ix} is the Q-value of action x taken by agent a_i. In Lines 9 and 10, $0 < \alpha < 1$ is the learning rate.

When finishing learning Q-values, a_i and a_j (Line 13) cooperate to find the optimal actions for both of them, where $match(x, y)$ is a function which is used to test whether the actions x and y that are taken by a_i and a_j, respectively, are matched. An action is only matched with its reverse action, such as that $form\_\prec$ is only matched with $form\_\succ$. Therefore, a_i and a_j have to cooperate to find the actions, which can be matched together and make the sum of their Q-values maximum.

Algorithm 2. Candidates selection of each agent

```
1  for each a_i ∈ A do
2      Candidates_i ← ∅;
3      for each Δ_k ∈ Tokens_i do
4          statistics of Δ_k.owner;
5      end for
6      if ∃ # of same Δ_k.owner > thre_1 and
7          Δ_k.owner ∉ Neig_i^≿ ∨ Neig_i^≻ ∨ Neig_i^≺ then
8          Candidates_i ← Candidates_i ∪ {Δ_k.owner};
9      end if
10     if ∃ # of same Δ_k.owner < thre_2 and
11         Δ_k.owner ∈ Neig_i^≿ ∨ Neig_i^≻ ∨ Neig_i^≺ then
12         Candidates_i ← Candidates_i ∪ {Δ_k.owner};
13     end if
14 end for
```

Algorithm 2 illustrates the reasoning aspect of each agent for selecting a group of agents to initilise the self-organisation process. As described in Section 2, each agent has not only the tokens it currently holds but also all the previous tokens incoming and outgoing through it. Then, each agent uses the local information provided by the tokens to choose candidates. Firstly, from Lines 3 to 5, agent a_i identifies the owner of each token stored in a_i's token list, $Tokens_i$, and counts the number of tokens from each owner. On the one hand, if the number of tokens from one owner exceeds a predefined threshold and this owner is not a neighbour of a_i, this owner will be added into the candidates set (Lines 6-9). This can be explained that if an agent is not a neighbour of agent a_i but often delegated tasks to a_i in the former task allocation processes, a_i might want to adapt the

relation with this agent. On the other hand, if the number of tokens from one owner is lower than another predefined threshold and this owner is a neighbour of a_i, this owner will be also appended into the candidates set (Lines 10-13). This can be explained that if an agent, which is a neighbour of a_i, delegated very few tasks to a_i in previous task allocation processes, then a_i might also want to alter the relation with this agent.

According to the description, our self-orgnisation mechanism in an agent network is based on only local information which is represented by tokens. In addition, both agents, a_i and a_j, employ the multiagent Q-learning method to learn optimal actions. In this way, our mechanism enables every pair of agents to independently learn the Q-value for taking any of the available actions towards altering their relation, and the two agents jointly select the actions which can be matched together and maximise the sum of their Q-values. In this manner, our mechanism could make the pair of agents be treated fairly, when they decide to change their relation.

4 Experiment and Analysis

In this section, the effectiveness of our self-organisation mechanism is demonstrated through experimental evaluation. We first describe the experimental setup and thereafter present the experimental results and analysis.

4.1 Experimental Setup

To objectively exhibit the effectiveness of our self-organisation mechanism, *Learn-Adapt*, we compare our mechanism with two other methods, namely *Central* and *K-Adapt* [10], which are depicted as follows.

1. *Central*: This is an ideal centralised task allocation mechanism in which there is an external omniscient central manager that maintains information about all the agents and tasks in the network. The central manager is able to interact with all the agents in the network without cost. Thereby, all task allocations are only one step direct allocation through the central manager. This method is not practical or robust, but it can be used as an upper bound of the performance of an organisation in our experiment.
2. *K-Adapt*: This method was proposed by Kota et al. [10], which utilised meta-reasoning approach to adapt the relation between two agents. This mechanism is somewhat biased from the agent which launches the relation adaptation process towards the agent which is requested to accessorily adaptation relation.

In this experiment, the agent organised network is generated by using the Small World network [13], in which most neighbours of an agent are connected to each other. Nevertheless, the approach presented by [13] deals with only one relation between agents in the Small World network. We, thus, modify the approach to accommodate multiple relations by randomly changing the relation between two neighbouring agents. Moreover, in order to control the number of resources an agent could hold, a parameter called *Resource Probability* (RP) is utilised, such

that an agent is assigned a resource with probability RP. Hence, with the increase of RP, agents could possess more resources. For simplicity, tasks are created by randomly generating resource and computation capacity requirements, and each task is randomly distributed to one of the agents in the network. Finally, the evaluated criteria is $Profit_{NET}$ (Equation 1), obtained by both *K-Adapt* and our *Learn-Adapt* in a percentage format with the maximum network profit gained by the *central* mechanism. For clarity, the values of parameters which are exploited in this experiment and their meanings are listed in Table 2.

Table 2. Parameters Setting

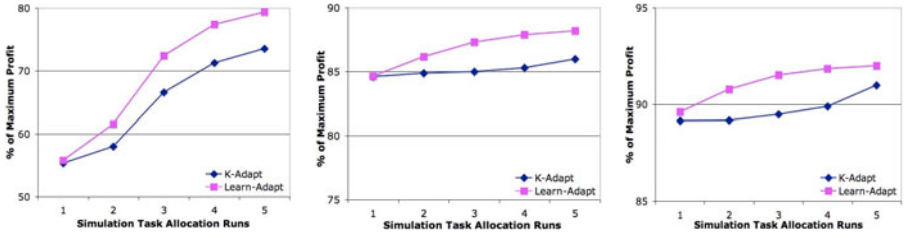
Parameters	Values	Explanations
n	$50 \sim 250$	The number of agents
deg	$4 \sim 10$	The average number of neighbours
RP	$0.1 \sim 0.6$	Resource Probability
m	15000	The number of tasks
α	0.2	Learning rate
ϵ	0.4	Action selection distribution probability
k	100	Learning rounds
$thre_1$, $thre_2$	2, 5	Thresholds for choosing agents to adapt

4.2 Experimental Results and Analysis

Figure 2 demonstrates the percentage profits obtained by both *K-Adapt* and *Learn-Adapt* with different resource probabilities (RP), compared with the maximum profit which is gained by *Central*. The number of agents in the network, n, is fixed at 50 and the average number of neighbours of each agent, deg, is set to 6. The x-aix represents the number of simulation task allocation runs and each run consists of 15000 tasks. It can be seen that *Learn-Adapt* performs consistently better than *K-Adapt* in all situations. In Figure 2(a) ($RP = 0.1$), with more task allocation runs, the difference between *Learn-Adapt* and *K-Adapt* is gradually increasing. This is because when each agent has very few resources, agents have to allocate tasks to others. Thus, an effective network structure could reduce agents' communication cost and management cost, and, further, raise the profit of the entire network. In this case, a smarter mechanism could bring better performance through generating a more effective network structure.

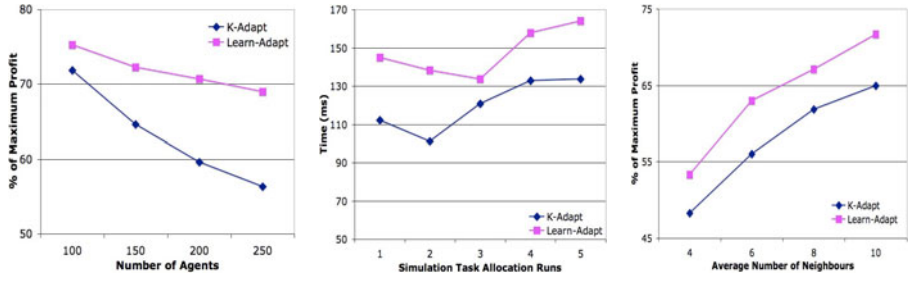
With the increase of resource probability (Figures 2(b) and 2(c)), both *K-Adapt* and *Learn-Adapt* could achieve better performance. This can be explained that with higher resource probability, each agent would have more resources and, thus, could fulfill more tasks by itself. It should also be noted that the difference between *Learn-Adapt* and *K-Adapt* narrows as the resource probability rises. This is because when agents become more homogeneous, a smarter method cannot correspondingly bring more profit for the network.

Furthermore, we estimated both *Learn-Adapt* and *K-Adapt* in other three aspects as well, which include their performance in different network scales ($n =$



(a) Resource Probability (RP)=0.1 (b) Resource Probability (RP)=0.3 (c) Resource Probability (RP)=0.6

Fig. 2. Relative Profits of *K-Adapt* and *Learn-Adapt* with Different Resource Probabilities (RP)



(a) Different Number of Agents (b) Time Consumption (c) Different Number of Neighbours

Fig. 3. Other Performance Indices of *K-Adapt* and *Learn-Adapt*

$100 \sim 250, deg = 6, RP = 0.1$), their overhead ($n = 250, deg = 6, RP = 0.1$) and their adaptability ($n = 250, deg = 4 \sim 10, RP = 0.1$). Figures 3(a), 3(b) and 3(c) show the experimental results of the three aspects, respectively. In Figure 3(a), it can be found that with the network scale increasing, i.e. more agents forming the network, the performance of both *Learn-Adapt* and *K-Adapt* decreases. This is because that, with the increase of network scale, each agent has to form a longer delegation chain to allocate its tasks, which will incur more communication cost and management load for maintaining subtasks. It should also be noticed, in Figure 3(a), that the performance of *K-Adapt* declines more sharply than our *Learn-Adapt*, which implies that the scalability of *Learn-Adapt* is better than *K-Adapt*. Figure 3(b) shows the time consumption of both *Learn-Adapt* and *K-Adapt* in each simulation task allocation run, which is constituted by 15000 tasks. As expected, the running of *K-Adapt* costs less time than *Learn-Adapt*. This can be explained that *Learn-Adapt* requires a time consuming learning round, as

other learning algorithms doing, and agents using *Learn-Adapt* have to take more reward evaluation attributes into account, as described in Subsection 3.1, in order to choose an optimal action to change their relations. However, the gap of time consumption between *learn-Adapt* and *K-Adapt* is only in milliseconds level, which should be acceptable in many real cases. Figure 3(c) displays that with the average number of neighbours of each agent rising, the performance of both *Learn-Adapt* and *K-Adapt* improves. Since more neighbours can effectively reduce the length of delegation chain of each subtask, communication cost and management cost for maintaining subtasks of each agent could be lowered as well, which will lead to the improvement of entire network profit.

In summary, the performance of our method is around 80% $\sim$ 90% of the upper bound centralised allocation method, and on average 5% better than the *K-Adapt* method in small sized network (50 agents). In larger agent networks, although the performance of both methods reduces, *Learn-Adapt* still achieves relatively higher performance than *K-Adapt*. Thereby, it has been proven that our learning method can achieve higher performance for self-organisation than meta-reasoning to some extent, and it is believed in many scenarios that the improvement of performance and scalability deserves a little growth of time consumption.

5 Conclusion

This paper introduced a decentralised self-organisation mechanism which aims to adapt structural relations among agents in a network to achieve efficient task allocation. By using this mechanism, a pair of agents can independently evaluate each available action and jointly make a decision about taking an action to change their relation. Since this mechanism is decentralised and continuous over time, it meets the principles of self-organisation defined by Serugendo et al. [1]. We also empirically demonstrated that the performance of our mechanism approaches to the centralised method and consistently outperforms *K-Adapt*.

This research can be exploited for task allocation in many complex systems where resources are distributed and agents are highly autonomous, such as agent-based grid systems, service-oriented computing and wireless sensor networks (WSN). For example, our mechanism could be applied for optimising packet routing in WSN, since packets are somewhat similar as tokens in our model. Although each packet in WSN has a pre-specified destination which is different from a token, our mechanism could still work with a few minor modifications. Other potential application domains include automatic systems which are capable of self-management, e.g. e-commerce and P2P information retrieval.

In the future, we plan to improve and test our mechanism in a dynamic environment where agents can leave and join at any time step, and new types of resources might be obtained or lost by agents over time. In addition, another interesting stream of future work is that three or more agents might jointly make a decision with regard to changing relations among them.

References

[1] Serugendo, G.D.M., Gleizes, M.P., Karageorgos, A.: Self-organization in multi-agent systems. The Knowledge Engineering Review 20(2), 165–189 (2005)
[2] Mathieu, P., Routier, J.C., Secq, Y.: Principles for dynamic multi-agent organizations. In: Kuwabara, K., Lee, J. (eds.) PRIMA 2002. LNCS (LNAI), vol. 2413, pp. 109–122. Springer, Heidelberg (2002)
[3] Horling, B., Benyo, B., Lesser, V.: Using self-diagnosis to adapt organizational structures. In: AGENTS 2001, Montreal, Quebec, Canada, pp. 529–536 (May 2001)
[4] Hoogendoorn, M.: Adaptation of organizational models for multi-agent systems based on max flow networks. In: IJCAI 2007, Hyderabad, India, pp. 1321–1326 (January 2007)
[5] Kamboj, S., Decker, K.S.: Organizational self-design in semi-dynamic environments. In: AAMAS 2007, Honolulu, Hawai'i, USA, pp. 1228–1235 (May 2007)
[6] Gaston, M.E., des Jardins, M.: Agent-organized networks for dynamic team formation. In: AAMAS 2005, Utrecht, Netherlands, pp. 230–237 (July 2005)
[7] Glinton, R., Sycara, K., Scerri, P.: Agent organized networks redux. In: AAAI 2008, Chicago, Illinois, USA, pp. 83–88 (July 2008)
[8] Abdallah, S., Lesser, V.: Multiagent reinforcement learning and self-organization in a network of agents. In: AAMAS 2007, Honolulu, Hawai'i, USA, pp. 172–179 (May 2007)
[9] Kota, R., Gibbins, N., Jennings, N.R.: Decentralised structural adaptation in agent organisations. In: AAMAS 2008 Workshop on Organized Adaption in Multi-Agent Systems, Estoril, Portugal, pp. 1–16 (May 2008)
[10] Kota, R., Gibbins, N., Jennings, N.R.: Self-organising agent organisations. In: AAMAS 2009, Budapest, Hungary, pp. 797–804 (May 2009)
[11] Xu, Y., Scerri, P., Yu, B., Okamoto, S., Lewis, M., Sycara, K.: An integrated token-based algorithm for scalable coordination. In: AAMAS 2005, Utrecht, Netherlands, pp. 407–414 (July 2005)
[12] Gomes, E.R., Kowalczyk, R.: Dynamic analysis of multiagent q-learning with e-greedy exploration. In: ICML 2009, Montreal, Canada, pp. 369–376 (June 2009)
[13] Watts, D.J., Strogatz, S.H.: Collective dynamics of 'small-world' networks. Nature 393, 440–442 (1998)