Agent Division and Fusion for Task Execution in Undependable Multiagent Systems

Yifeng Zhou and Yichuan Jiang*, *IEEE Senior Member*

Distributed Intelligence and Social Computing Laboratory,
School of Computer Science and Engineering, Southeast University, Nanjing 211189, China.
**Corresponding author, email: yjiang@seu.edu.cn*

***Abstract*—In multiagent systems, agents with limited capacity often cooperate in order to accomplish various types of tasks. Due to the openness of multiagent systems, agents may run the risk of their cooperations to accomplish tasks because of some involved undependable agents. The state of the art for handling this problem concentrates on the phase of task allocation, which aims to allocate tasks to more dependable agents (*task allocation-oriented*). In fact, accomplishing a task in a multiagent system requires two phases: i) *task allocation*, and ii) *task execution*. Hence, this paper, on the contrary, focuses on managing the phase of task execution to guarantee the performance of task accomplishment (*task execution-oriented*). To reduce the performance loss caused by undependable agents, the proposed *agent division and fusion* mechanism enables agents to autonomously divide themselves into sub-agents for executing tasks with different risks (more resources will be assigned to the sub-agent with lower risk in task execution); then the sub-agents can also fuse together and make a re-division to fit the current task environments. This work is also expected to be able to complement the existing state of the art (task allocation-oriented) for guaranteeing the performance of task accomplishment in undependable multiagent systems from task execution-oriented perspective.**

***Keywords*-Multiagent system; agent division; agent fusion; undependable; task allocation; task execution.**

I. INTRODUCTION

Accomplishing tasks is a common duty of multiagent systems (MASs) [1-2]. In MASs, agents often jointly accomplish tasks through cooperation, since each agent may have limited types and amount of resources and thus cannot satisfy the requirement of a task solely [3]. The agent who has a waiting task to accomplish should find other agents with suitable resources to obtain commitment to jointly execute the task [4]. Hence, accomplishing a task in MASs often requires two phases: i) *task allocation*, and ii) *task execution*.

In recent years, many task allocation mechanisms in MASs have been proposed, which intend to improve the system performance for accomplishing tasks by allocating tasks to the most suitable set of agents to execute [5-8]. There is an underlying assumption in most of these works: the agents in the system are all cooperative, i.e., all the agents are initiative and dependable during the task accomplishing process [8]. In fact, this type of multiagent system which is widely concerned can be called "dependable multiagent systems" (D-MASs).

In recent research, "undependable multiagent systems" (U-MASs) has been also studied [1-2]. In U-MASs, some undependable agents are involved [2][9], which may take malicious behaviors, e.g., manipulating their resource status information and breaking the contract, in the cooperation with other agents to accomplish tasks [1-2][4][10]. For this reason, the system performance may decrease significantly, because of the failure of accomplishing tasks. To solve this problem, task allocation models incorporating guaranteeing mechanisms, e.g., the reputation model [4][11] and reward/punishment mechanism [1], have been proposed, which are expected to increase the dependability of task accomplishment by allocating tasks to more dependable agents [1][4]. These models concentrate on the first phase of task accomplishment (*task allocation*), thus can be called *task allocation-oriented* model for U-MASs.

On the contrary, in this paper we concentrate on the second phase of task accomplishment (*task execution*) in U-MASs. In D-MASs, there are some researches where agents are able to make autonomous adaptation of themselves to achieve a better performance during the process of task execution [12-15]. These works provide inspiration for us to propose *task execution-oriented* mechanism for U-MASs.

In U-MASs, an agent may run different levels of risk when it cooperates with different agents to execute tasks. If the agent executes the waiting tasks as it does in a D-MASs, the utility it gained may suffer greatly, because this agent may spend much time and all its resources on executing tasks which actually may not be able to be accomplished. To solve this problem, an *agent division and fusion* mechanism (AgentD&F) is proposed in this paper, which enables agents to autonomously divide themselves into paired sub-agents for executing tasks with different risks (more resources will be assigned to the sub-agent with lower risk in task execution, i.e., with more dependable cooperators); and the sub-agents can also fuse together and make a re-division to fit the current task environments[1]. Hence, the allocated tasks can be naturally executed with different priorities according to their

[1] Agent fusion in this paper indicates the *sibling-fusion*, which can be conducted by a pair of sub-agents that are derived from the same original agent. *Cross-fusion*, which means the fusion of sub-agents with different original agents, will be discussed in our future work.

different levels of risks; the task with the lower risk will be executed with the higher priority.

Through experimental evaluations, the performance of the presented AgentD&F mechanism for U-MASs has been validated. It can improve the utilities the system gained by decreasing the failure rate of task execution.

The remainder of this paper is organized as follows. Section 2 presents some typical related works. Section3 introduces the task accomplishing environment in the U-MASs. Section 4 presents the details of the AgentD&F mechanism. Section 5 presents experimental evaluations that validate the effectiveness of the AgentD&F mechanism. Finally, Section 6 concludes our paper and discusses the future works.

II. Related Work

A. Task Allocation in Undependable Multiagent Systems

Traditional works of task allocation in U-MASs often incorporate some guaranteeing mechanisms, e.g. the reputation model [1][4][11], the reward/punishment method [1][4] and mechanism design [10], to improve the system performance of task accomplishment.

Weerdt et al. proposed a task allocation approach using mechanism design, which can guarantee the dependability of agents' cooperation by incentivizing agents to report their status correctly [10]. Sonnek et al. presented reputation mechanism to estimate reliability ratings of agents; and then proposed several reputation-based task scheduling algorithms to achieve efficient task allocation [11].

In our previous works [1][4], we proposed task allocation models for U-MASs with simplex and multiplex network structures respectively. In [1], a reputation model, which considered not only the dependability of agents but also the negotiation path, had been proposed to guarantee the dependability of task allocation in the U-MASs with simplex network structure. Then in [4], we extended our task allocation model to fit the U-MASs with multiplex network structure, which considered an additional aspect, the dependability of the network layer, in the reputation model.

Overall, these approaches introduced above concentrated on the *task allocation* phase of task accomplishment in U-MASs, whereas they did not consider improving the system performance from the *task execution* perspective. In this paper, we aim to guarantee the system performance from the *task execution* perspective, and the mechanism proposed is expected to be able to be combined with the traditional *task allocation-oriented* approaches for U-MASs to achieve a much better performance.

B. Self-Organization in Dependable Multiagent Systems

Self-organizing mechanisms can enable agents to adapt their configurations to fit the current environment of the MASs in order to achieve better performance of task accomplishment [12-14].

Mathieu et al. introduced three basic principles of self-organizing multiagent systems, which are relation adaptation, resource exchange and cloning/spawning [12]. Kota et al. presented a decentralized structure adaptation mechanism, which can enable agents to adapt their relations to accomplish tasks more efficiently [13]. Ye et al. [14] proposed an integrative self-organizing mechanism, which considered all the three basic principles of self-organizing multiagent system introduced by Mathieu et al. [12].

These works can improve the task accomplishing performance of the systems effectively in D-MAS. However, the situation of the failure of task execution which may be caused by the behaviors of undependable agents in U-MASs has not been taken into account. Despite this, these works provide inspiration for us to propose the agent division and fusion mechanism (AgentD&F) for U-MASs from *task execution-oriented* perspective. AgentD&F can be seen as a new variant of the principle, cloning/spawning, introduced by Mathieu et al.[12].

III. Task Accomplishing Environment in U-MASs

A. The System Model

Definition 1: Multiagent system (MAS). *A multiagent system is given by <A, E>, where A is the set of agents, and $\forall <a_i,a_j> \in E$ indicate the existed relations between agent a_i and a_j. Agents in the MAS can cooperate to accomplish tasks through their relations.*

Definition 2: Agent. *An agent a_i can be characterized by 1) E_i, the relations with its neighbors N_{a_i}, 2) R_{a_i}, its resources for executing tasks, and 3) Q_{a_i}, its task queue waiting for execution. Thus, a_i is represented by a tuple $< E_i, R_{a_i}, Q_{a_i}>$.*

1. *E_i is the set of relations between agent a_i and its neighbors N_{a_i}.*
2. *$R_{a_i} = (r^1_{a_i}, r^2_{a_i},\ldots, r^k_{a_i})$, where $r^k_{a_i}$ is the number of type k resources of agent a_i.*
3. *$Q_{a_i} = (Q_{a_i}(0), Q_{a_i}(1),\ldots, Q_{a_i}(l))$, where $Q_{a_i}(l)$ is the l^{th} task in the task queue Q_{a_i} of agent a_i, and l is the length of task queue Q_{a_i}.*

Definition 3: Task. *A task, T, arriving to the multiagent system can be represented by $<R_T, U_T, t_T>$, where R_T is the requirement of a set of resources for accomplishing task T, U_T is the utility that can be gained from the successful accomplishment of T and t_T is the time limit for accomplishing task T. The successful accomplishment of the task T can be achieved if the following conditions are satisfied:*

1. *The resource requirement R_T is satisfied, i.e., $R_T \subseteq \bigcup_{\forall a_x \in A_T} R_{a_x}$, where A_T is the set of agents who cooperate to execute the task T.*
2. *The time limit t_T for accomplishing T is satisfied, i.e. $t_{T'} \le t_T$, where $t_{T'}$ is the real executing time of T.*

Definition 4: Task accomplishment in MAS. *A task T can be accomplished by performing the following two steps:*

1. *Task allocation: first, seeking for the set of agent A_T, which satisfies $\forall a_i, a_j \in A_T$, $P_{ij} \subseteq E$, and $R_T \subseteq \bigcup_{\forall a_x \in A_T} R_{a_x}$, where P_{ij} is the negotiation path between a_i and a_j; second, assigning the predetermined sub-tasks (each*

sub-task is a part of the resource requirement of T) to corresponding agents, i.e., inserting the sub-tasks into their waiting task queues.

2. *Task execution: when all the sub-tasks of T allocated to the set of agent A_T are executed, i.e., there is no sub-task of T that still exists in the task queues of A_T waiting for execution, the execution of task T is over.*

Definition 5: Dependable and undependable agents for task execution in U-MASs. *The dependability of an agent is determined by two aspects of its behaviors in accomplishing tasks:*

1) Resource status reporting in task alloction.

- *A dependable agent reports its real status of resource to the system or other agents in task allocation* [1][4].
- *An undependable agent fabricates its status of resource in task allocation, e.g. over-reporting* [1][4].

2) The strategy in executing allocated tasks.

- *A dependable agent, a_i, executes the tasks in its task queue Q_{a_i} in order, i.e. $\forall$ $Q_{a_i}(x)$, $Q_{a_i}(y) \in Q_{a_i}$ and $x<y$, $Q_{a_i}(y)$ must be executed after $Q_{a_i}(x)$.*
- *An undependable agent, a_i, may manipulate the task executing order, i.e. it allows the situation that, if $x<y$, $Q_{a_i}(y)$ can be executed before $Q_{a_i}(x)$.*

B. The Objective

Agents cooperate to jointly accomplish tasks in order to gain utilities from tasks arriving to the system. As demonstrated in previous section, each task can provide a defined utility if it is accomplished successfully. But if the task is not accomplished because of the unsuccessful execution or the overtime execution, the utility cannot be obtained.

We define the utility rate (*UR*) to reflect the effective-ness of the system for accomplishing tasks. Utility rate is defined as follows.

Definition 6: Utility rate. *Let {T} be the set of tasks arriving to the system, {T}' is the set of tasks that are accomplished, and {T}'⊆{T}. The Utility rate (UR) is calculated by*

$$UR_{\{T\}} = \sum\nolimits_{\forall T_i \in \{T\}'} U_{T_i} \Big/ \sum\nolimits_{\forall T_j \in \{T\}} U_{T_j} \tag{1}$$

where U_{T_i} is the utility of task T_i.

Therefore, the objective of task accomplishment in undependable multiagent systems (U-MASs) is to **maximize the utilities obtained from the set of tasks arriving to the systems**.

As undependable agents involved, the tasks may have different probabilities to be accomplished when they are allocated to different agents. Thus, to satisfy this objective, the limited resources of the agents should be preferentially accessed to execute the tasks with higher probability to be accomplished.

We propose the agent division and fusion mechanism, which enables agents to autonomously divided themselves into paired sub-agents for executing tasks with different risks; and the sub-agents can also fuse together and make re-division to fit the current task environments. Hence, the tasks can be naturally executed with different priorities according to their different levels of risks. Thus, the utility loss caused by undependable agents can be reduced.

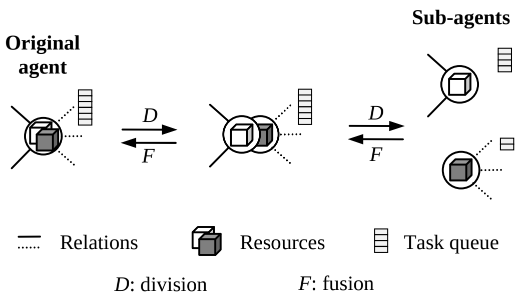

Figure 1. An example of agent division and fusion.

IV. Agent Division and Fusion

Agent division and fusion (AgentD&F) mechanism enables the agent (called *original agent*) to divide itself into two sub-agents and also enable the sub-agents to fuse together to make a re-division. Fig. 1 gives an example of agent division and fusion.

Each sub-agent obtains a part of the relations, resources, and the allocated tasks of the original agent. Hence, agent division and fusion requires the following aspects.

- *Relation division and fusion*: the set of relations of an original agent a_i, E_i, can be divided into two groups: low-risk group, E_i^+, and high-risk group, E_i^-,where $E_i = E_i^+$ E_i^-. The high-risk and low-risk groups are composed of the relations with neighbors with relatively lower and higher dependability (see Definition 8). The sub-agents will have E_i^+ and E_i^-, respectively. The relation fusion is the inverse process of relation division.
- *Resource division and fusion*: the resources of the original agent a_i, R_{a_i}, should be also divided into two parts: $R_{a_i}^+$ and $R_{a_i}^-$, according to the division results of the relations. More resources will be assigned to the sub-agent with low-risk group of relations. The resource fusion is the inverse process of resource division.
- *Task queue division and fusion*: the allocated task queue of the original agent a_i, Q_{a_i}, should be divided into $Q_{a_i}^+$ and $Q_{a_i}^-$, according to the division results of relations. If a task is allocated from the relations in E_i^+, the task will be divided into the sub-agent with E_i^+, and vice versa. The task queue fusion is the inverse process of task queue division.

Thus, agent division and fusion can be defined as:

$$\overbrace{a_i < E_i, R_{a_i}, Q_{a_i} >}^{original\ agent} \underset{F}{\overset{D}{\rightleftarrows}} (\underbrace{a_i^+ < E_i^+, R_{a_i}^+, Q_{a_i}^+ >}_{riskless\ sub-agent}, \underbrace{a_i^- < E_i^-, R_{a_i}^-, Q_{a_i}^- >}_{risky\ sub-agent}) \tag{2}$$

By performing agent division, two sub-agents can be produced from an original agent. The sub-agent, a_i^+, which carries the low-risk relations E_i^+, is called the *riskless sub-agent*; and the sub-agent, a_i^-, which carries the high-risk relations E_i^-, is called the *risky sub-agent*. And by performing agent fusion, the two sub-agents can join together to be the original agent.

To implement agent division and fusion, there are two important problems need to be solved: *when and how to perform agent division and fusion*.

A. Activation of agent division and fusion (When?)

➢ *Activation of agent division*

Whether to perform agent division by an agent is determined by two aspects: the *division desire* derived from the current task executing situation of the agent and the *division threshold* of this agent.

In the following, we first give some preliminary definitions.

Definition 7: Failure rate of task execution. *Let ξ_i represent the failure rate of task execution of agent a_i, then ξ_i is defined as:*

$$\xi_i = (h_i - h_i^{succ}) / h_i \tag{3}$$

where h_i is the total number of tasks agent a_i has executed ($h_i > 0$), and h_i^{succ} is the number of tasks accomplished successfully.

Definition 8: Dependability of agents and tasks. *Let h_{ij} be the total number of tasks allocated by agent a_j then executed by a_i, h_{ij}^{succ} be the corresponding number of tasks accomplished succesfully. The dependability of agent a_j evaluated by a_i is defined as:*

$$\sigma_{ij} = 0.5 + 0.5 \cdot (h_{ij}^{succ} - (h_{ij} - h_{ij}^{succ})) / h_{ij} \tag{4}$$

Note that $h_{ij}>0$, if $h_{ij}=0$, $\sigma_{ij} = 0.5$.
The dependability of a task T allocated by a_j to a_i is defined as:

$$\tau(T) = \sigma_{ij} \tag{5}$$

The *division desire* of an agent is influence by the circumstance of the task execution. On one hand, the failure rate of past tasks executed by the agent should be first considered. If the failure rate is higher, the agent should be more willing to perform the mechanism of agent division to improve the task execution performance. Then on the other hand, the influence (utility loss) on the performance of task execution caused by agent division also needs to be considered. Such influence has been often considered by the self-organizing mechanism in D-MASs [13][15]. In order to decrease the utility loss, agent division should be performed when there are few tasks waiting in the task queue, and the tasks with higher dependability tend to be at the queue tail (according to the following theorem).

Let Q_{a_i} be the current task queue of a_i. Then we have the following theorem.

Theorem 1. *It is assumed that the relation division of the agent division will be correct and the allocated tasks in Q_{a_i} can be executed just satisfying their corresponding time limits. Then if the dependable tasks are at the task queue tail, the utility loss caused by agent division will be the least.*

Proof sketch. Performing agent division, the task queue Q_{a_i} will be divided into $Q_{a_i}^+$ and $Q_{a_i}^-$. Because of the resource division, more execution time for each task will be required. Then if the tasks in Q_{a_i} can just satisfy the corresponding time limits, the tasks at the queue tail of Q_{a_i} will have higher probability to be accomplished after division because of their larger time limits. Then According to assumption of the correctness of the relation division of the recent agent division, the set of dependable tasks $\{T_i\}^+$ can be accomplished if their time limits are satisfied, and the set of undependable tasks $\{T_i\}^-$ cannot be accomplished even if their time limits are satisfied. Then if $\{T_i\}^+$ can be at the tail of Q_{a_i}, the expected utility loss will be minimized.

Definition 9: Division desire. *Let Q_{a_i} be the current task queue of a_i. The division desire of agent a_i, DE_i, is defined as:*

$$DE_i = \xi_i \cdot \frac{1}{|Q_{a_i}|+1} \cdot g(Q_{a_i}) \tag{6}$$

where

$$g(Q_{a_i}) = \begin{cases} \dfrac{\sum_{x=0}^{|Q_{a_i}|-1} \tau(Q_{a_i}(x)) \cdot x}{|Q_{a_i}| \cdot (|Q_{a_i}|-1)/2}, & |Q_{a_i}| \geq 1 \\ 1, & |Q_{a_i}| = 0 \end{cases} \tag{7}$$

In Equation (7), x is the index of the task in the task queue, and $\tau(Q_{a_i}(x))$ is the dependability of the task $T(Q_{a_i}(x))$ calculated by Equation (5).

Therefore, if $DE_i > Thes_i$, where $Thes_i$ is the division threshold of agent a_i, agent division will be performed by a_i immediately.

➢ *Activation of agent fusion*

With the execution of some tasks after agent division, the sub-agents can evaluate whether they can take advantage of the recent division for task execution; if the experiences indicate that the recent division cannot take effect recently, the agent fusion will be performed by the sub-agents. (the sub-agents we mentioned here are with the same original agent.)

The conditions for agent fusion are presented as below:

$$h_i^{+succ} \big/ h_i^+ < 1 - \xi_i - \varepsilon \tag{8}$$

$$h_i^{-succ} \big/ h_i^- > 1 - \xi_i + \varepsilon \tag{9}$$

where h_i^{+succ} and h_i^{-succ} are the numbers of tasks executed successfully by the riskless and risky agent, a_i^+ and a_i^-, respectively; h_i^+ and h_i^- are the total numbers of the tasks executed by a_i^+ and a_i^-, respectively; ξ_i is the failure rate of task execution calculated by (3) in recent division; ε is the tolerance parameter set by the agent ($\varepsilon > 0$). When the conditions in (8) and (9) are satisfied, agent fusion will be performed immediately.

B. *Agent division and fusion (How?)*

When an agent decides to perform agent division or fusion, there are three aspects of division and fusion should be conducted: *relation* division and fusion, *resource* division and fusion, and *task queue* division and fusion. We present the approaches that show how to conduct each type of division and fusion in the following.

➢ *Relation division and fusion*

The relation division plays the most important role in agent division compared with the resources division and task queue division, since it determines whether the cooperators (neighbors) with the agent can be effectively divided into groups for executing tasks with relative lower risk and higher risk, respectively.

For relation division, we use a k-means clustering approach [16] to divide the set of relations of the original agent into two sets. Each relation can be a node in the set, and the "*location*" of the relation is the dependability of its corresponding agent, for example, σ_{ij} can represent the location of the relation e_{ij} between agent a_i and a_j. Two initial cluster centers in the clustering approach are set to 0.0 and 1.0, respectively. The algorithm for relation division is presented in Algorithm 1. Moreover, with the relation division, the direct neighbors of the agent have been also divided into N_i^+ and N_i^-.

Relation fusion in agent fusion is an inverse process of relation division. It can be simply presented by $E_i = E_i^+ + E_i^-$. When sub-agents decide to make agent fusion, the relation fusion will be firstly conducted. Due to space limitation, we do not present the details of relation fusion here.

➢ *Resource division and fusion*

After the relation division, the resource division needs to be conducted to determine how many resources should be assigned to the riskless and risky sub-agents. To reduce the risk of task execution and improve the system performance, more resources of the original agent will be assigned into the riskless sub-agent, such that more utilities are expected to be obtained from the dependable tasks since they will have a high probability to be accomplished and will be executed with higher priority.

Hence, performing resource division should directly refer to the results of relation division. Two aspects of the results of relation division need to be taken into account: 1) the numbers of relations in E_i^+ and E_i^-; 2) the difference of dependability between E_i^+ and E_i^-. The former aspect can naturally determine the quantity of tasks each sub-agent will be allocated; the more relations the sub-agent has, the more resources will be assigned to this agent. The latter aspect can reflect the dependability difference of the allocated tasks; hence if the larger difference of the dependability of the relations respectively in E_i^+ and E_i^-, more resources will be assigned to the riskless sub-agent.

Algorithm 1. Relation Division.

```
/* k: the number of clusters; E_i: the set of relations of agent a_i; Dis_{j-cp} and Dis_{j-co}: the distance between e_ij and the cluster center cp and co in the k-means clustering approach, respectively. */
Input: $k=2$, and $E_i$.
$E_i^+=\{\}$, $E_i^-=\{\}$;
$e_{ix} = \arg\max_{e_{ix}\in E_i} \sigma_{ix}$; $e_{iy} = \arg\min_{e_{iy}\in E_i} \sigma_{iy}$;
$cp=1.0$; $co=0.0$; /*set initial cluster center*/
$cp'=-1$; $co'=-1$;
While (($cp$!=$cp'$) or ($co$!=$co'$)) do:
    $cp'=cp$; $co'=co$;
    For $\forall e_{ij}\in E_i$, do:
        $Dis_{j-cp} = |\sigma_{ij} - cp|$;
        $Dis_{j-co} = |\sigma_{ij} - co|$;
        If ($Dis_{j-cp} < Dis_{j-co}$), then: $E_i^+ = E_i^+ \quad e_{ij}$;
        Else: $E_i^- = E_i^- \quad e_{ij}$;
    $cp = (\sum_{\forall e_{ij}\in E_i^+} \sigma_{ij}) / |E_i^+|$
    $co = (\sum_{\forall e_{ik}\in E_i^-} \sigma_{ik}) / |E_i^-|$
Output: $E_i^+$ and $E_i^-$.
```

Let $|E_i^+|$,$|E_i^-|$and $|E_i|$ be the number of relations in the riskless sub-agent, the risky sub-agent and the original agent, respectively ($|E_i|=|E_i^+|+|E_i^-|$), σ_{ij} represent the dependability of the relation e_{ij} between agent a_i and a_j. The resources division is defined as:

$$R_{a_i}^+ = R_{a_i} \cdot (|E_i^+| / |E_i| + \omega \cdot (\frac{\sum_{\forall e_{ij}\in E_i^+} \sigma_{ij}}{|E_i^+|} - \frac{\sum_{\forall e_{ik}\in E_i^-} \sigma_{ik}}{|E_i^-|})) \quad (10)$$

$$R_{a_i}^- = R_{a_i} \cdot (|E_i^-| / |E_i| - \omega \cdot (\frac{\sum_{\forall e_{ij}\in E_i^+} \sigma_{ij}}{|E_i^+|} - \frac{\sum_{\forall e_{ik}\in E_i^-} \sigma_{ik}}{|E_i^-|})) \quad (11)$$

where ω is the parameter to adjust the preference for resource assignment, and $\omega > 0$.

Resource fusion is also an inverse process of resource division, which can be represented by $R_{a_i} = R_{a_i}^+ + R_{a_i}^-$. Due to space limitation, we do not present the details of resource fusion here.

➢ *Task queue division and fusion*

After the relation division, the task queue division can be conducted according to the division results of relations of the original agent (actually the division results of the direct neighbors, N_i^+ and N_i^-).

Let $T(Q_{a_i}(x))$ be a task in the waiting task queue of agent a_i, where x is the index of the task in the queue. If $T(Q_{a_i}(x))$, is executed through the cooperation with agent a_j, and $a_j \in N_i^+$, then this task will be assigned to the riskless sub-agent a_i^+, i.e., this task will be inserted into the riskless sub-agent's task queue, $Q_{a_i}^+$, and vice versa. Finally, by conducting task queue division, the task queue of a_i, Q_{a_i}, will be divided into $Q_{a_i}^+$ and $Q_{a_i}^-$, which are the task queues of the corresponding sub-agents, respectively.

Task queue fusion is the most important aspect of agent fusion, since it will influence the task execution order of the allocated tasks. In order to guarantee the performance of task accomplishment, we get the following theorem to guide the task queue fusion.

Let $Q_{a_i}^+$ and $Q_{a_i}^-$ be the waiting task queues of the riskless sub-agent a_i^+ and the risky sub-agent a_i^-, respectively, and Q_{a_i} be the waiting task queue of the expected fused agent.

Theorem 2. *It is assumed that the relation division of the recent agent division is correct and the allocated tasks in $Q_{a_i}^+$ and $Q_{a_i}^-$ can be executed just satisfying their corresponding time limits. Then, if the task queue fusion is conducted by inserting $Q_{a_i}^+$ and $Q_{a_i}^-$ into Q_{a_i} in order, the optimal task queue fusion can be achieved.*

Proof sketch. According to the assumption of the correctness of the relation division of the recent agent division, the tasks in $Q_{a_i}^+$ are expected to be accomplished if they are executed satisfying their time limit, while the tasks in $Q_{a_i}^-$ cannot be accomplished caused by some agents' undependable behaviors. Hence, the expected maximum utility equals $\sum_{\forall T_i \in Q_{a_i}^+} U_{T_i}$. Combining $Q_{a_i}^+$ *and* $Q_{a_i}^-$together into Q_{a_i} causes the arrangement of the tasks, which will lead in the situation that the execution of some tasks cannot satisfy their time limits. Hence, only by inserting $Q_{a_i}^+$ and $Q_{a_i}^-$ into Q_{a_i} in order can make the execution of all the tasks in $Q_{a_i}^+$ satisfy their time limit, and then can be accomplished. Therefore, the expected utility can equals $\sum_{\forall T_i \in Q_{a_i}^+} U_{T_i}$. □

V. Experimental Validation

A. Experimental Settings

To validate the effectiveness of the proposed agent division and fusion mechanism (AgentD&F) for guaranteeing the performance of task accomplishment in undependable multiagent systems, the index, *utility rate* (UR) introduced in Definition 6 is employed. UR indicates how many utilities are finally gained from the set of tasks arriving to the system.

Accomplishing a task in MASs requires two phases: i) *task allocation*, and ii) *task execution*. The presented AgentD&F mechanism can be conducted simultaneously in the task execution phase when agents are executing the allocated tasks. Then, in the task allocation phase, the allocation model used by the system to allocate tasks to agents is introduced in the following, which accords with the *manager/contractor* architecture [1] of task allocation in MASs.

- A task, *T*, arriving to the system can be firstly allocated to a randomly selected agent, a_i, who acts as the manager for this task.
- The manager agent, a_i, should then request the agents in its direct neighbor set N_i for contractors. The set of agents who responds to the request is NC_i.
- If the requirements of the resource and time limit of *T* can be satisfied by the set of agents $\{a_i, NC_i\}$, the task *T* is allocated successfully. Otherwise, the allocation process will return to the manager selection step.

The initial network of the U-MASs is constructed by a random network model [17], in which 100 agents are included. The probability of relation construction for any randomly selected agents is set to 0.15. The division threshold for each agent is set randomly, ranging from 0 to 0.05; and the tolerance parameter, ε, for the activation of agent fusion (see Equation 8 and 9) and the preference parameter, ω, for resource division (see Equation 10 and 11) are both set to 0.1 in the experiments. The two types of undependable behaviors of agents (See Definition 5 in Section 3) are both involved in the system. Note that each experiment in this section comprises 100 runs to obtain the average results.

B. Results and Analyses

Fig. 2 shows the experimental results of the utility rate (UR) and the total utility of task accomplishment with different number of tasks arriving to the system.

Fig. 2(a) shows the test results of utility rate of the tasks arriving to the system. We find that incorporating our mechanism into task execution, the utility rate of the system can be improved remarkably; and with the increase of the number of tasks, the performance of our mechanism can be better. This indicates that agent division and fusion mechanism can learn and fit the environment well. For more clear understanding, we have shown the corresponding results of total utility of the system in Fig. 2(b). A higher utility rate of task accomplishment in Fig 2(a) implies that a higher total utility of the system can be gained.

The reasons for such results are as follows. Without our mechanism, agents will cooperate with other agents with equal priority. i.e., an agent will execute the allocated tasks in accordance with the allocated sequence and with all its resources. However, with the involved undependable agents, many tasks may be not able to be accomplished. There will be a large amount of utility loss. Then on the contrary, after evaluating the dependability of relations and neighbors, agent division and fusion mechanism enables agents to cooperate with others with different priorities by dividing itself into two sub-agents. More resources can be spent to cooperate with more dependable neighbors to accomplish tasks. Hence, the utility loss caused by undependable agents will be reduced. Moreover, with more experiences of task execution, agents can have more accurate evaluations of the dependability of its relations and neighbors; then by conducting agent fusion to make re-division, the system performance can be better. This is also the reason of the evolving property of our mechanism.

Fig. 3 shows the test results of failure rate of tasks caused by two types of undependable behaviors of agents in the system (See Definition 5 in Section 3): (i) resource information fabrication (type-1), and (ii) waiting task queue manipulation (type-2). Fig. 3(a) and (b) show the results of from *task execution-oriented* perspective enable agents to change their configurations autonomously in the task execution process in order to improve the task accomplishing performance. To reduce the utility loss caused by undependable agents, the presented *agent division and fusion* mechanism first enables agents to divide themselves into two sub-agents to cooperate with agents of different dependability to execute tasks (more resources of the original agent will be assigned to the sub-agent with more dependable neighbors); then it can also enables the sub-agents to fuse together to make a re-division to fit the current task environments. Through experimental evaluations, we indicate that incorporating the agent division and fusion mechanism into task execution process in U-MASs, the failure rate of tasks can be largely reduced; then the utility rate of tasks gained by the system can be improved.

In future work, we will devise a cross-fusion mechanism which can enable sub-agents from different original agents to fuse together to obtain further improvement of task accomplishing performance. Moreover, a comprehensive trust model used by the agents to achieve more accurate dependability evaluation of relations and neighbors will be incorporated into the agent division and fusion mechanism. Finally, we will try to incorporate this mechanism into our task allocation model [1] for U-MASs to seek for a larger guaranteeing effect on task accomplishing performance in U-MASs.

ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China (No.61170164, No. 61472079), the Program for Distinguished Talents of Six Domains in Jiangsu Province (No.2011-DZ023), and the Funds for