Comparing Repair-Task-Allocation Strategies in MAS

Hisashi Hayashi
System Engineering Laboratory, Corporate Research & Development Center, Toshiba Corporation,
1 Komukai-Toshiba-cho, Saiwai-ku, Kawasaki, 212-8582, Japan

Keywords: Multi-Agent Systems, Coordination Mechanism, Distributed Task Allocation, Emergency Repair.

Abstract: Many distributed systems can be regarded as multi-agent systems (MASs) where some agents are connected to a network but located in different places. We consider severe situations where many causes of future agent failures in MASs are found simultaneously and consecutively owing to large-scale disasters. If a cause of future agent failure is not removed within a limited time, there is a high possibility that one of the agents will stop working. In order to find effective strategies that reduce the number of actual agent failures, we compare some repair-task-allocation strategies for MASs where sensing agents find causes of future agent failures and manager agents communicate with one another to allocate repair tasks to action-execution agents.

1 INTRODUCTION

In many multi-agent systems (MASs), even if an agent stops working, most of the other agents continue to work and cover the task of the disabled agent unless some important agents break down. Normally, relatively few causes of future agent failures are found simultaneously and it is not difficult to remove them before some agents actually fail. However, in the event of large-scale disasters, many causes of future agent failures are found simultaneously and consecutively, in which case, the whole MAS will stop functioning if they are not fixed efficiently using limited resources. Therefore, it is important to effectively allocate the resources necessary to repair them.

In this paper, we consider some scenarios where disaster events repeatedly occur, which create causes of future agent failures simultaneously and consecutively. In such scenarios, we compare some repair-task-allocation algorithms for MASs to find a way of reducing the number of damaged agents so that the MAS continues to function as a whole. Although some fundamental repair-task-allocation algorithms are compared in (Beaumont and Chaib-draa 2007), the maximum number of causes of future agent failures in their test scenarios is 10. On the other hand, in our disaster scenarios, we consider much more severe situations where the maximum number is of the order of hundreds. As discussed in Section 6, this difference leads to completely different conclusions. Although cooperation between agents was ineffective for reducing the number of failures in the test scenarios of (Beaumont and Chaib-draa 2007), we predict that cooperation is vital for allocation of limited resources within a limited time when many causes of future agent failures are created simultaneously and consecutively owing to large-scale disasters.

As discussed in (Choi, Brunet, and How 2009; Macarthur et al. 2011; Rahimzadeh, Khanli, and Mahan 2015), task-distribution algorithms are roughly divided into two kinds of algorithms: centralized algorithms and distributed algorithms. In centralized algorithms, a single manager agent collects information from its child agents and allocates tasks to them. On the other hand, in distributed algorithms, multiple manager agents communicate with one another to allocate the tasks to their child agents. Many existing task-allocation algorithms are centralized algorithms. However, distributed task-allocation algorithms are attracting attention because they are expected to be fault-tolerant: the breakdown of one manager agent does not mean the total failure of the MAS. One of our aims is to compare these two kinds of algorithms in the repair-task-allocation scenarios. Because execution of a repair action sometimes fails, it is natural to replan and reallocate the repair task. We also evaluate centralized and distributed repair-task-allocation algorithms that include replanning capabilities.

Hayashi H.
Comparing Repair-Task-Allocation Strategies in MAS.
DOI: 10.5220/0006094800170027
In *Proceedings of the 9th International Conference on Agents and Artificial Intelligence (ICAART 2017)*, pages 17-27
ISBN: 978-989-758-219-6

There is more related work on task allocation. In (Rahimzadeh, Khanli, and Mahan 2015), the probabilities of future agent failures are considered when allocating tasks to agents. However, the algorithm does not consider repairing. In (Guessoum et al. 2010), some backup agents are created in case of emergency. However, as pointed out in (Rahimzadeh, Khanli, and Mahan 2015), the cost of backup agents is high when additional hardware is needed and it takes time to copy the agents dynamically. The algorithm does not consider repairing either. In (Okimoto et al. 2015), considering future agents' failure, robust agent teams are created. The idea is to prepare more agents than needed. Again, the cost is high and repairing is not considered. Coallition formation of first responders in disaster relief is also researched in (Ramchurn et al. 2010).

Applications of multi-agent task allocation range from disaster relief (Chapman et al. 2009; dos Santos and Bazzan 2011; Nair et al. 2002; Pujol-Gonzalez 2015; Ramchurn et al. 2010; Ramchurn et al. 2016; Suárez, Quintero, de la Rosa, 2007), computer games (Dawe 2013; Straatman et al. 2013), and coordination of robots (Mi et al. 2014; Vallejo et al. 2009) to command and control for combat ships (Beaumont and Chaib-draa 2004; Beaumont and Chaib-draa 2007; Brown and Lane 2000; Brown et al. 2001; Young 2005). Our repair-task-allocation is closely related to the task allocation problems of combat ships and disaster relief where tasks with hard deadlines such as threat removal and civilian rescue are allocated to teams.

This paper is organized as follows. In Section 2, we define the MAS architecture for repair-task allocations. In Section 3, we define five algorithms for task repairing. In Section 4, we explain the detailed settings for simulation. In Section 5, we show the simulation results. In Section 6, we discuss the simulation results, comparing them with the simulation results reported in (Beaumont and Chaib-draa 2007). Section 7 is devoted to the conclusion.

2 MAS ARCHITECTURE FOR REPAIR-TASK ALLOCATIONS

We consider a MAS for repair-task allocations that is composed of multiple unit MASs including sensing agents, action-execution agents, and a manager agent: sensing agents detect causes of future agent failures, action-execution agents fix causes of future agent failures using limited resources, and manager agents communicate with one another to allocate repair tasks to action-execution agents. In this section, we define unit MASs and the agents that belong to unit MASs. We define the functions of unit MASs as agents because each function is often deployed on different hardware and becomes out of order independently.

As shown in Figure 1, a **unit MAS** is a MAS comprising 0 or more **sensing agents**, 0 or more **action-execution agents**, and 1 **manager agent**. When a sensing agent senses a cause of a future agent failure, it reports the information to the manager agent in the same unit MAS. When receiving the information of a cause of a future agent failure, the manager agent allocates the repair task to an action-execution agent that belongs to the same unit MAS or allocates the repair task to the manager agent of another unit MAS if there are multiple unit MASs and their manager agents are connected by the network.

When allocated a repair task, the action-execution agent will execute a repair action consuming one resource. Execution of a repair action will succeed or fail according to the predefined probability. Unless a cause of a future agent failure is removed by a repair action, one of the agents will stop functioning according to the predefined probability.

When a unit MAS has sensing agents and action-execution agents, the unit MAS can sense and remove causes of future agent failures without the help of the other unit MASs as shown in Figure 1. In this case, each unit MAS is **independent** and it is unnecessary to connect each unit MAS through the network. Because of its simplicity, the repair-task-allocation algorithm for independent unit MASs will be compared with other algorithms as a baseline algorithm.

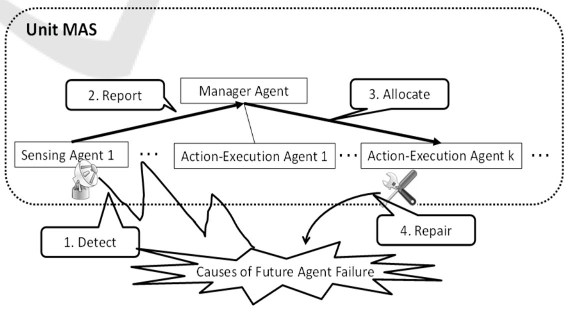

Figure 1: A unit MAS.

As shown in Figure 2, when the manager agents of unit MASs are connected by the network, sensing and repairing can be done in different unit MASs. We expect that repair-task allocations will be more effective when the manager agents are connected.

When the MAS has a **centralized** architecture, as shown in Figure 3, only one manager agent (**top**

manager agent) works as a leader and allocates all the repair tasks to unit MASs. On the other hand, when the MAS has a **distributed** architecture as shown in Figure 2, different manager agents become leaders of task allocation for different repair tasks. We expect that the distributed MAS algorithm is more robust than the centralized algorithm because in the distributed architecture, even if a leader becomes out of order, another manager agent becomes a leader. We will compare the repair-task-allocation algorithms that use these MAS architectures.

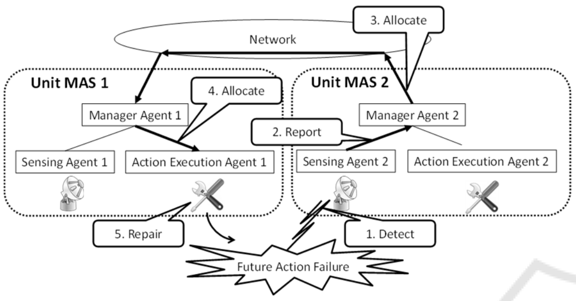

Figure 2: Distributed architecture.

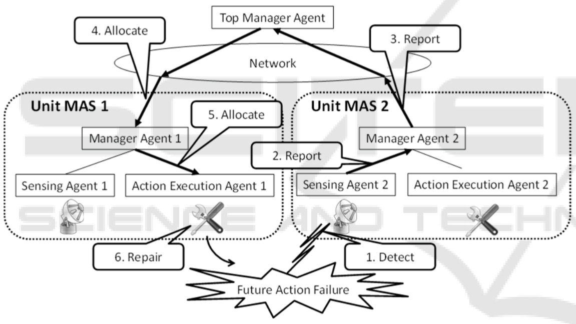

Figure 3: Centralized architecture.

3 ALGORITHMS

In this section, we introduce five algorithms for repair-task allocations: the independent unit MAS algorithm (baseline algorithm), the centralized algorithm, the distributed algorithm, the centralized algorithm with replanning, and the distributed algorithm with replanning. In order to make these algorithms for repair-task-allocations, we use well-known techniques.

In the independent unit MAS algorithm, the manager agent of each unit MAS decides which repair action to execute without exchanging information with other unit MASs. In the centralized algorithm, only one manager agent (top manager agent) works as a leader and allocates repair tasks to unit MASs. In the distributed algorithm, when a sensing agent detects a cause of a future agent failure, the manager agent in the same unit MAS works as a leader and selects a unit MAS for the repair-task. Note that different manager agents become leaders for different repair tasks in the distributed algorithm. In the centralized algorithm with replanning, only the top manager agent reallocates unsuccessful repair tasks to unit MASs. On the other hand, in the distributed algorithm with replanning, when an action-execution agent fails to execute a repair action, the manager agent in the same unit MAS works as a leader and reallocates the unsuccessful repair task to a unit MAS.

We expect that the centralized algorithm and the distributed algorithm can allocate repair tasks more effectively than the independent unit MAS algorithm because unit MASs communicate with one another. We also expect that the distributed algorithm is more robust than the centralized algorithm that is weak with respect to the failure of the top manager agent. We also evaluate the centralized algorithm with replanning and the distributed algorithm with replanning because it is natural and effective to reallocate the repair task when the repair task is not completed successfully.

3.1 Independent Unit MAS Algorithm

In the independent unit MAS algorithm, the manager agents of different unit MASs do not communicate with one another. Therefore, as in Figure 1, when a sensing agent senses a cause of a future agent failure, the action-execution agent in the same unit MAS tries to repair it without the help of the other unit MASs. Although this algorithm does not require network connections among unit MASs, multiple action-execution agents in different unit MASs might try to execute repair actions for the same cause of a future agent failure, which leads to unnecessary consumption of resources.

Algorithm 1 (Independent Unit MAS Algorithm)
The sensing agents, the manager agent and the action-execution agents in each unit MAS work as follows if they are alive:

- *Algorithm of Sensing Agents*

1. *When a sensing agent detects a new cause of a future agent failure, it reports the information to the manager agent in the same unit MAS if the manager agent is alive.*

- *Algorithm of Manager Agents*

1. *When the manager agent M receives the information of a new cause of a future agent failure C from a sensing agent in the same unit*

MAS, the manager agent M selects and reserves an action-execution agent E for the repair task R of C, if E exists, such that E is alive, the number of resources of E is more than 0 and E is not reserved for another cause of a future agent failure.

2. *When it becomes possible for the reserved action-execution agent E to start executing the repair action A for the reserved repair task R, if E is alive, the manager agent M in the same unit MAS orders E to execute A and erases the reservation information.*

- *Algorithm of Action-Execution Agents*

1. *When receiving an execution order of the repair action A, the action-execution agent E executes A, decrements 1 resource whether the result of A is a success or a failure, and reports the result to its manager agent.*

3.2 Centralized Algorithm

A centralized algorithm is often used for task allocations in general. In the centralized algorithm, as shown in Figure 3, only one manager agent (top manager agent) allocates repair tasks to the manager agents of unit MASs based on the contract net protocol (Smith 1980), which is a kind of auction. Although there are many criteria to select a unit MAS, we select the unit MAS that can start repairing first. Because of the network connection, we expect that the centralized algorithm is more effective than the independent unit MAS algorithm.

Algorithm 2 (Centralized Algorithm) *The sensing agents, the manager agent and the action-execution agents in each unit MAS work as follows if they are alive:*

- *Algorithm of Sensing Agents*
  Same as the algorithm of sensing agents in Algorithm 1.
- *Algorithm of Manager Agents*

1. *When the manager agent M receives the information of a new cause of a future agent failure C from a sensing agent in the same unit MAS, the manager agent M forwards the information of C to the top manager agent T if T is alive. Otherwise, it works in the same way as the first step of the algorithm of manager agents in Algorithm 1.*
2. *When the manager agent M receives an allocation of a repair task R from the top manager agent, M selects and reserves an action-execution agent E for R in the same way as the first step of the algorithm of manager agents in Algorithm 1.*
3. *Same as the second step of the algorithm of manager agents in Algorithm 1.*

- *Algorithm of the Top Manager Agent*

1. *When the top manager agent T receives the information of a new cause of a future agent failure C from a manager agent, T asks each alive manager agent M whether the unit MAS U of M can be in charge of the repair task R of C and how quickly an action-execution agent of U can start the repair action of R. Then, T allocates R to the manager agent of the unit MAS U2 such that an action-execution agent of U2 can start the repair action of R the quickest.*

- *Algorithm of Action-Execution Agents*
  Same as the algorithm of action-execution agents in Algorithm 1.

3.3 Distributed Algorithm

In the distributed algorithm, as shown in Figure 2, when a sensing agent detects a cause of a future agent failure, the manager agent in the same unit MAS works as a leader like the top manager agent of the centralized algorithm, selects a unit MAS based on the contract net protocol (Smith 1980), and allocates the repair-task to the manager agent of the selected unit MAS. However, unlike the centralized algorithm, different manager agents become leaders for different repair tasks. Therefore, the distributed algorithm is expected to be more robust for agent failures than the centralized algorithm.

Algorithm 3 (Distributed Algorithm) *The sensing agents, the manager agent and the action-execution agents in each unit MAS work as follows if they are alive:*

- *Algorithm of Sensing Agents*
  Same as the algorithm of sensing agents in Algorithm 1.
- *Algorithm of Manager Agents*

1. *When the manager agent M receives the information of a new cause of a future agent failure C from a sensing agent in the same unit MAS, M asks each alive manager agent, selects a unit MAS U and allocates the repair task of C to the manager agent of the unit MAS U in the same way as the first step of the algorithm of the top manager agent in Algorithm 2.*
2. *When the manager agent M receives an allocation of a repair task R from the manager agent of another unit MAS, M selects and*

reserves an action-execution agent E for R in the same way as the first step of the algorithm of manager agents in Algorithm 1.

3. *Same as the second step of the algorithm of manager agents in Algorithm 1.*

- *Algorithm of Action-Execution Agents*
  Same as the algorithm of action-execution agents in Algorithm 1.

3.4 Centralized Algorithm with Replanning

In the centralized algorithm with replanning, when execution of a repair action results in failure, the top manager agent reselects a unit MAS and reallocates the repair task to the manager agent of the selected unit MAS. Because the success rate of repair actions is not 100%, it is expected that replanning will decrease the number of agent failures.

Algorithm 4 (Centralized Algorithm with Replanning) *The sensing agents, the manager agent and the action-execution agents in each unit MAS work as follows if they are alive:*

- *Algorithm of Sensing Agents*
  Same as the algorithm of sensing agents in Algorithm 1.
- *Algorithm of Manager Agents*

1. *Same as the first step of the algorithm of manager agents in Algorithm 2.*
2. *Same as the second step of the algorithm of manager agents in Algorithm 2.*
3. *Same as the third step of the algorithm of manager agents in Algorithm 2.*
4. *When the manager agent M receives the result of repair-action execution for the repair task R from an action-execution agent in the same unit MAS, if the result is a failure, M reports the result of R as a failure to the top manager agent.*

- *Algorithm of the Top Manager Agent*

1. *Same way as the first step of the algorithm of the top manager agent in Algorithm 2.*
2. *When the top manager agent T receives the result of a failure for the repair task R from a manager agent, T asks each alive manager agent, selects a unit MAS U, and reallocates R to one of the manager agents in the same way as the first step of the algorithm of the top manager agent in Algorithm 2.*

- *Algorithm of Action-Execution Agents*
  Same as the algorithm of action-execution agents in Algorithm 1.

3.5 Distributed Algorithm with Replanning

In the distributed algorithm with replanning, when an action-execution agent fails to execute a repair action, the manager agent in the same unit MAS reselects a unit MAS and reallocates the task to the manager agent of the selected unit MAS. It is expected that replanning will decrease the number of agent failures.

Algorithm 5 (Distributed Algorithm with Replanning) *The sensing agents, the manager agent and the action-execution agents in each unit MAS work as follows if they are alive:*

- *Algorithm of Sensing Agents*
  Same as the algorithm of sensing agents in Algorithm 1.
- *Algorithm of Manager Agents*
  1. *Same as the first step of the algorithm of manager agents in Algorithm 3.*
  2. *Same as the second step of the algorithm of manager agents in Algorithm 3.*
  3. *Same as the third step of the algorithm of manager agents in Algorithm 3.*
  4. *When the manager agent M receives the result of repair-action execution for the repair task R from an action-execution agent in the same unit MAS, if the result is a failure, this manager agent M reallocates R to one of the manager agents in the same way as the first step of the algorithm of manager agents in Algorithm 3.*
- *Algorithm of Action-Execution Agents*
  Same as the algorithm of action-execution agents in Algorithm 1.

4 SIMULATION SETTINGS

In this section, we explain the details of simulation settings used to compare and evaluate the algorithms defined in the previous section. We set typical values of unit MASs, considering our target application that is closely related to the application of (Beaumont and Chaib-draa 2007). In 4.1, we show the number of agents and resources in unit MASs. In 4.2, we show the performances of sensing and repairing of each unit MAS. In 4.3, we show the consecutive occurrence patterns of disaster events that create multiple causes of future agent failures simultaneously. As stressed in the introductory section, we predict that cooperation among agents becomes very effective in these severe occurrence patterns of disaster events, which is completely different from the result of (Beaumont and Chaib-draa 2007) where the numbers of causes of future agent failures in their test scenarios are small.

4.1 Numbers of Agents and Resources in Unit MASs

As shown in Table 1, we use 7 kinds of unit MASs: UMAS 0, …, UMAS 6, which are typical unit MASs of our target application. The numbers of UMAS 0, …, UMAS 6 are 1, 1, 2, 2, 4, 8, 8. The total number of these unit MASs is 26 (=1+1+2+2+4+8+8). Each UMAS has exactly one manager agent and the total number of manager agents is 26. Each UMAS has exactly one sensing agent except UMAS 0 and the total number of sensing agents is 25.

Table 1: The number of unit MASs, agents, and resources.

Unit MAS Type	# of Deployed Unit MASs	# of Manager Agents	# of Sensing Agents	# of Action-Execution Agents	# of Initial Resources of Each Action-Execution Agent
UMAS 0	1	1	0	0	-
UMAS 1	1	1	1	0	-
UMAS 2	2	1	1	0	-
UMAS 3	2	1	1	4	18
UMAS 4	4	1	1	2	12
UMAS 5	8	1	1	1	24
UMAS 6	8	1	1	1	3

Because high-performance unit MASs are costly in general, considering the balance, we use a smaller number of high-performance unit MASs and a larger number of low-performance unit MASs. We introduce the performance of each unit MAS in the next subsection. Although we conducted simulations for different numbers of these unit MASs, we observed similar simulation results and we do not show them in this paper. Note that the aim of this paper is to find when each algorithm is more effective than others. Simulation under different combinations of unit MASs was not helpful for this purpose.

The numbers of action-execution agents in UMAS 0, …, UMAS 6 are 0, 0, 0, 4, 2, 1, 1. The total number of action-execution agents is 32 (=2*4+4*2+8*1+8*1). An action-execution agent cannot execute more than one repair action in parallel but multiple action-execution agents can execute repair actions at the same time. The numbers of initial resources that each action-execution agent in UMAS 3, …, UMAS 6 has are 18, 12, 24, 3. The total number of initial resources is 456 (=2*4*18+4*2*12+8*1*24+8*1*3).

UMAS 1 and UMAS 2 do not have action-execution agents, which means that the causes of agent failures found by the sensing agent of UMAS 1 or UMAS 2 need to be repaired by the action execution agents of UMAS 3, …, UMAS 6. UMAS 0 does not have sensing agents or action-execution agents. It has only one manager agent that works as the top manager agent when using the centralized algorithm or the centralized algorithm with replanning.

4.2 Performance of Sensing and Repairing

Table 2 shows performances of unit MASs in terms of sensing and repairing. We set the typical performances of each unit MAS considering a specific application. Sensing agents in UMAS 1, …, UMAS 6 can start detecting causes of agent failure respectively from 360, 180, 72, 43.2, 18, 18 seconds before the expected time of agent's breakdown. The sooner the sensing agent detects a cause of a future agent failure, the higher the performance is, which means that performance of the sensing agent in UMAS 1 is the best. The probability of detecting causes of future agent failures is 90%.

Action-execution agents in UMAS 3, …, UMAS 6 can start repairing from 36, 18, 10.8, 10.8 seconds before the expected time of an agent failure. The sooner the action-execution agent can start repairing, the higher the performance is, which means that performance of the action-execution agent in UMAS 3 is the best. The success probability of repairing is 80%. When an action-execution agent starts repairing x seconds before the expected time of agent's breakdown, the time of repairing will be x/2.5 seconds. We assume a situation where a cause of agent failure approaches the target agent at constant speed and the action-execution agent sends the resource for a repair to the cause of future agent failure at constant speed.

Table 2: The performance of sensing and repairing.

Unit MAS Type	Performance of Sensing		Performance of Repairing		
	Time (secs) to Start Detecting a Cause before an Agent Failure	Prob. of Detecting Causes of Future Agent Failures (%)	Time (secs) to Start Repairing before an Agent Failure	Time for Repairing (secs) when Starting the Repair x secs before an Agent Failure	Success Prob. of Repairs (%)
UMAS 1	360	90	-	-	-
UMAS 2	180	90	-	-	-
UMAS 3	72	90	36	x/2.5	80
UMAS 4	43.2	90	18	x/2.5	80
UMAS 5	18	90	10.8	x/2.5	80
UMAS 6	18	90	10.8	x/2.5	80

4.3 Occurrence Patterns of Disasters

Table 3 summarizes the occurrence patterns of disaster events and causes of future agent failures. In our simulation scenarios, when a disaster event occurs, a cause of a future agent failure is created every second. The total number of causes of future agent failures created by a disaster event is 30. Disaster events repeatedly happen 10 times, and the interval between disaster events is 1 hour. Note that the number of action-execution agents is 32, which is also the maximum number of the repair actions that can be executed in parallel. This number is closer to the number of causes of future agent failures created by a disaster event. Therefore, the repairing capability of the MAS is close to the limitation.

As summarized in Table 4, we use two different selection rules of agents: the random selection rule of agents and the concentrated selection rule of agents. When the random selection rule of agents is applied, if a cause of a future agent failure is not removed by a repair action, one of the agents is randomly selected. The selected agent becomes out of order with the probability of 90% in 1800 seconds. When the concentrated selection rule of agents is applied, the top manager agent (the manager agent of UMAS 0) is selected with the probability of 10% and one of the other agents is randomly selected if the top agent is not selected. We expect that the concentrated selection rule affects the centralized algorithm.

Table 3: Simulation settings.

# of Disaster Events	# of Causes of Future Agent Failures per Disaster Event	Interval between Disaster Events (hours)	Interval between Occurrences of Causes of Future Agent Failures in a Disaster Event (secs)	Time from an Occurrence of a Cause to the Agent Failure (secs)	Prob. of Agent Failures when the Causes Are Not Removed (%)
10	30	1	1	1800	90

Table 4: Selection rules of agents.

Random Selection Rule of Agents	Concentrated Selection Rule of Agents
One agent is randomly selected.	The top manager agent is selected with the prob. of 10% and one agent is randomly selected otherwise.

5 SIMULATION RESULTS

This section shows the simulation results. We conducted simulations 1000 times using different random seeds for each algorithm and for each selection rule of agents. We show the results in terms of the number of agent failures and successful repairs. This section is composed of two subsections. In 5.1, we show the simulation results when using the random selection rule of agents. In 5.2, we show the simulation results when using the concentrated selection rule of agents.

5.1 When using the Random Selection Rule of Agents

This subsection shows the simulation results when using the random selection rule of agents. Figure 4 shows the average number of agent failures and Figure 5 shows the average number of successful repairs. Reducing agent failures is the top priority but it depends on successful repairs.

The centralized algorithm and the distributed algorithm are much better than the independent unit MAS algorithm because each cause of future agent failure is allocated to a unit MAS in the centralized algorithm and the distributed algorithm, which is not the case with the independent unit MAS algorithm. These two algorithms become much better when combined with a replanning capability because replanning covers unsuccessful repair actions.

The distributed algorithm with/without replanning is better than the centralized algorithm with/without replanning. However, the difference is slight. We expect that the difference would become bigger after the top manager agent becomes out of order. Therefore, in the next subsection, we focus on use of the concentrated selection rule of agents to increase the possibility of a failure of the top manager agent.

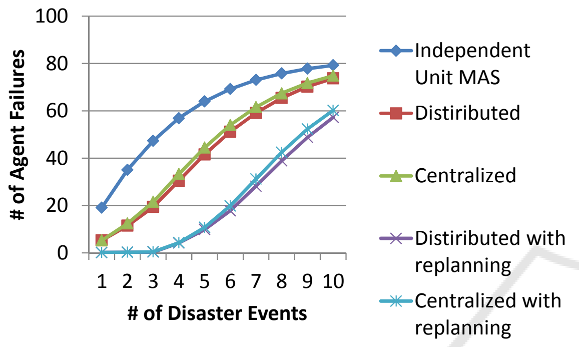

Figure 4: Average number of agent failures when using the random selection rule of agents.

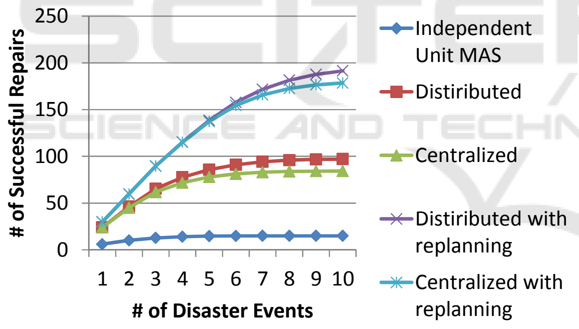

Figure 5: Average number of successful repairs when using the random selection rule of agents.

Figure 6 shows the probabilities of each algorithm getting the best results in terms of the numbers of agent failures. The sum of the probabilities is more than 100 % because multiple algorithms can get the best result in a trial. Unlike Figure 4, Figure 6 clearly indicates that the distributed algorithm with replanning is the best choice and the centralized algorithm with replanning is the second best choice.

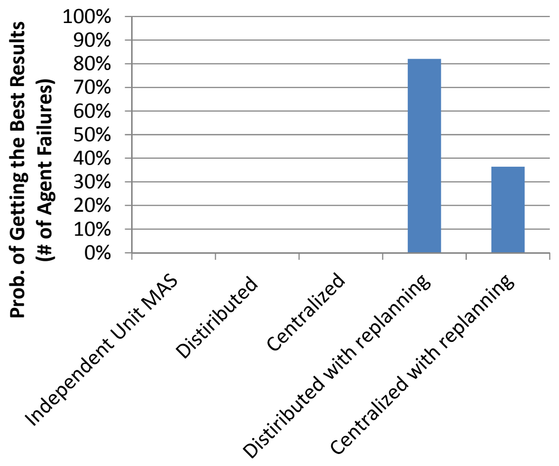

Figure 6: Probability of getting the best result in terms of the number of agent failures when using the random selection rule of agents.

5.2 When using the Concentrated Selection Rule of Agents

This subsection shows the simulation result when using the concentrated selection rule of agents. Figure 7 shows the average number of agent failures and Figure 8 shows the average number of successful repairs.

Now, the differences between the centralized algorithm with/without replanning and the distributed algorithm with/without replanning are clear. The distributed algorithm without replanning becomes better than the centralized algorithm without replanning from the second disaster event. The distributed algorithm with replanning becomes better than the centralized algorithm with replanning from the fifth disaster event. This is because the centralized algorithms work in the same way as the independent unit MAS algorithm after the failure of the top manager agent.

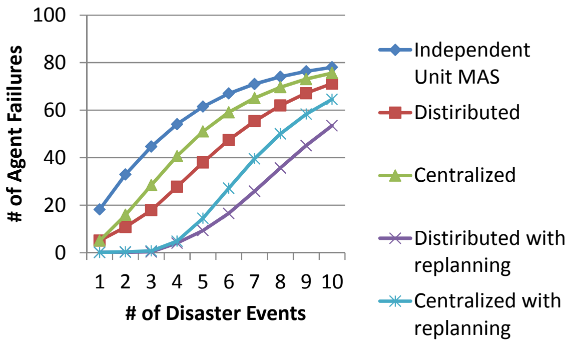

Figure 7: Average number of agent failures when using the concentrated selection rule of agents.

Figure 9 shows the probabilities of getting the best results in terms of the numbers of agent failures. Figure 9 clearly indicates that the distributed algorithm with replanning is the best choice.

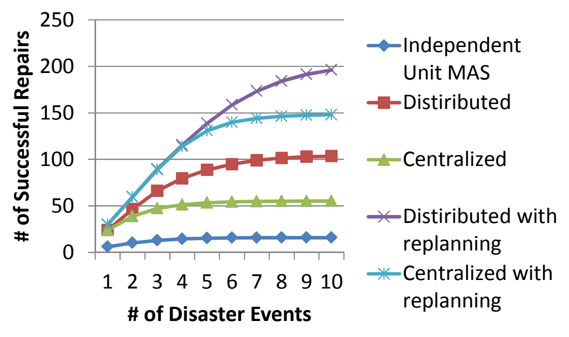

Figure 8: Average number of successful repairs when using the concentrated selection rule of agents.

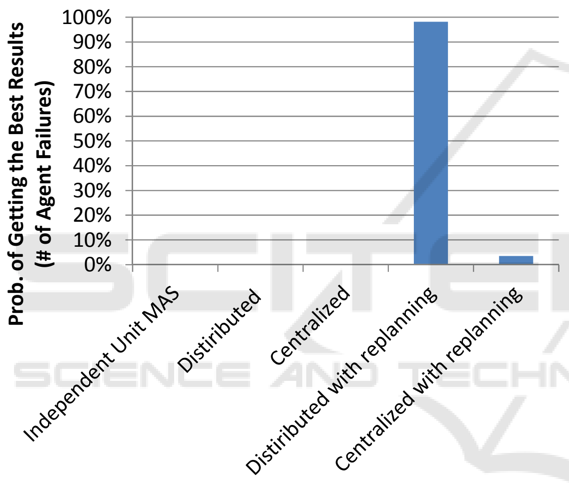

Figure 9: Probability of getting the best result in terms of the number of agent failures when using the concentrated selection rule of agents.

6 DISCUSSION

As mentioned in the introduction, some repair-task-allocation algorithms are compared in (Beaumont and Chaib-draa 2007), including a no-coordination algorithm (≒independent unit MAS algorithm), a zone defence coordination algorithm, a contract net algorithm, a simple centralized coordination algorithm, and another central coordination algorithm of (Brown and Lane 2001). In their simulation results, the no-coordination algorithm gives the best result in terms of the number of hits (≒successful repairs). Note that the independent unit MAS algorithm (≒no-coordination algorithm) always produces the worst results in our simulation results.

This is mainly because the total number of threats (≒causes of future agent failures) is small (1 to 10) in the scenario of (Beaumont and Chaib-draa 2007). This is also because each ship (≒unit MAS) tries to remove all the threats using many resources in the no-coordination algorithm whereas each threat is assigned to only one ship in the other algorithms and replanning is not an option even when it fails to remove the threat.

On the other hand, in our simulation scenario, 10 disaster events repeatedly happen, each disaster event creates 30 causes of future action failures, and the total number of causes of future agent failures is 300. In this case, the unit MASs cannot remove all the causes of future agent failures without coordination.

Also, in the independent unit MAS algorithm that uses many resources, unit MASs soon run out of resources at early stages and cannot continue repairing afterwards. Therefore, in our much severer disaster scenario, the independent unit MAS algorithm is the worst choice and coordination among unit MASs is vital. This is completely different from the simulation result of (Beaumont and Chaib-draa 2007) where they never run out of resources.

In addition, we found that replanning is very effective in this severe situation. Furthermore, we confirmed that in our simulation scenario, the distributed algorithm is better than the centralized algorithm and especially so when the concentrated selection rule is applied.

Because different scenarios might lead to different conclusions, it is important to conduct simulations considering our target applications in more detail. It is also important to evaluate more algorithms considering other situations such as network delay. We need to tackle these challenges as the next step.

7 CONCLUSIONS

In this paper, we evaluated and compared five algorithms for repair-task allocations in MASs that consist of multiple unit MASs by means of multi-agent simulation: the independent unit MAS algorithm, the centralized algorithm, the distributed algorithm, the centralized algorithm with replanning, and the distributed algorithm with replanning. We used severe simulation scenarios where causes of future agent failures are created simultaneously and consecutively. We conducted simulation 1000 times for each case and algorithm and confirmed the following in terms of the average number of agent failures and successful repairs:

- The centralized algorithm and the distributed algorithm are always better than the independent unit MAS algorithm in our simulation scenarios, which means that cooperation between unit MASs is very effective. This conclusion is completely different from the simulation results reported in (Beaumont and Chaib-draa 2007). This is because much more severe disasters are considered in our simulation scenarios where causes of future agent failures are created simultaneously and consecutively.
- The centralized algorithm and the distributed algorithm become even better when combined with replanning, which means that replanning is very effective in our simulation scenarios where repair actions sometimes fail.
- The distributed algorithm with/without replanning is better than the centralized algorithm with/without replanning when the random selection rule is applied although the difference is slight. However, the distributed algorithm with/without replanning becomes much better than the centralized algorithm with/without replanning when the concentrated selection rule is applied. This means that the distributed algorithm is effective for unbalanced occurrences of future agent failures and the centralized algorithm is not robust when the top manager agent is vulnerable.
  - In summary, the distributed algorithm with replanning is always the best and the independent unit MAS algorithm is always the worst in our severe simulation scenarios where hundreds of causes of future agent failures are created.

We also evaluated the algorithms from the view point of “the probabilities of getting the best results in terms of the numbers of agent failures” and confirmed the following:

- The distributed algorithm with replanning is clearly the best choice in any case in our simulation scenarios.
- The centralized algorithm with replanning is clearly the second best choice when the random selection rule of agents is applied.

In future work, we intend to consider the following two directions:

- We intend to evaluate the algorithms in more detail in our target application. For this purpose, we need to combine the MAS controller of our algorithms and the domain-specific simulator of our target application.
- We intend to evaluate more algorithms considering other situations. For example, sometimes the network speed between manager agents of different unit MASs might slow or the network might be cut off. In another example, the human manager of the unit MAS might correct the allocation of repair actions that the manager agent recommends.

REFERENCES

Beaumont, P. and Chaib-draa, B. “Multi-platform Coordination in Command and Control." In *Proceedings of the International Command and Control Research and Technology Symposium*, 2004.

Beaumont, P. and Chaib-draa, B. “Multiagent Coordination Techniques for Complex Environments: the Case of a Fleet of Combat Ships.” *IEEE Transaction on Systems, Man and Cybernetics-Part C*, 37(3), pp. 373-385, 2007.

Brown, C., and Lane, D. “Anti-Air Warfare Co-ordination - An Algorithmic Approach.” In *Proceedings of the International Command and Control Research and Technology Symposium*, 2000.

Brown, C., Fagan, P., Hepplewhite, A., Irving, B., Lane, D., and Squire, E. “Real-time Decision Support for the Anti-air Warfare Commander.” In *Proceedings of the International Command and Control Research and Technology Symposium*, 2001.

Chapman, A., Micillo, R. A., Kota, R. and Jennings, N. R. "Decentralised Dynamic Task Allocation: A Practical Game-theoretic Approach." In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, pp. 915–922, 2009.

Choi, H.-L., Brunet, L., and How, J. P. “Consensus-Based Decentralized Auctions for Robust Task Allocation.” *IEEE Transactions on Robotics* 25(4), pp. 912-926, 2009.

Dawe, M. "Beyond the Kung-Fu Circle: A Flexible System for Managing NPC Attacks.” In *Game AI Pro: Collected Wisdom of Game AI Professionals*, Chapter 28, pp.369-375, 2013.

dos Santos, F. and Bazzan, A. L. C. “Towards Efficient Multiagent Task Allocation in the RoboCup Rescue: a Biologically-Inspired Approach.” *Autonomous Agents and Multi-Agent Systems*, 22 (3), pp. 465-486, 2011.

Guessoum, Z, Briot, J. P., Faci, N., and Marin, O. “Toward Reliable Multi-Agent Systems: an Adaptive Replication Mechanism.” *Multiagent and Grid Systems*, 6(1), pp. 1-24, 2010.

Macarthur, K. S., Stranders, R., Ramchurn, S. D. and Jennings, N. R. “A Distributed Anytime Algorithm for Dynamic Task Allocation in Multi-Agent Systems.” In

Proceedings of the AAAI Conference on Artificial Intelligence, pp. 701-706, 2011.

Mi, Z., Yang, Y., Ma, H., and Wang, D. “Connectivity Preserving Task Allocation in Mobile Robotic Sensor Network.” In *Proceedings of the IEEE International Conference on Communications*, pp. 136-141, 2014.

Nair, R., Ito, T., Tambe, M., and Marsella, S. “Task Allocation in the Rescue Simulation Domain: a Short Note.” In *RobuCup2001*, LNAI 2377, Springer, pp. 751-754, 2002.

Okimoto, T., Schwind, N., Clement, M., Riberio, T., Inoue, K., and Marquis, P. “How to Form a Task-Oriented Robust Team.” In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, pp. 395-403, 2015.

Pujol-Gonzalez, M., Cerquides, J., Farinelli, A., Meseguer, P., and Rodriguez-Aguilar, J. A. “Efficient Inter-Team Task Allocation in RoboCup Rescue.” In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, pp. 413-421, 2015.

Rahimzadeh, F., Khanli, L. M., and Mahan, F. “High Reliable and Efficient Task Allocation in Networked Multi-Agent Systems.” *Autonomous Agent and Multi-agent Systems*, 29(6), pp.1023-1040, 2015.

Ramchurn, S. D., Farinelli, A., Macarthur, K. S., and Jennings, N. R. “Decentralized Coordination in RoboCup rescue.” *The Computer Journal*, 53(9):1447–1461, 2010.

Ramchurn, S. D., Wu, F., Jiang, W., Fischer, J. E., and Reece, S., Roberts, S., Rodden, T., Greenhalgh, C., and Jennings, N. R. "Human–agent Collaboration for Disaster Response." *Autonomous Agents and Multi-Agent Systems*, 30 (1). pp. 82-111, 2016.

Smith, R. G. “The Contract Net Protocol: High-level Communication and control in a Distributed Problem Solver.” *IEEE Transactions on Computers*, C-29(12), pp. 1104-1113, 1980.

Straatman, R., Verweij, T., Champandard, A., Morcus, R., and Kleve, H. “Hierarchical AI for Multiplayer Bots in Killzone 3.” Game AI Pro: Collected Wisdom of Game AI Professionals, Chapter 29, pp.377-390, 2013.

Suárez, S., Quintero, C., and de la Rosa, J. L. “Improving Tasks Allocation and Coordination in a Rescue Scenario.” In *Proceedings of the European Control Conference*, pp. 1498-1503, 2007.

Vallejo, D., Remagnino, P., Monekosso, D. N., Jiménez, L., and González, C. “A Multi-Agent Architecture for Multi-robot Surveillance.” In *Proceedings of the IEEE International Symposium on Parallel and Distributed Processing with Applications*, pp. 11-18, 2009.

Young, B. W. “Future integrated Fire Control.” In *Proceedings of the International Command and Control Research and Technology Symposium*, 2005.