Advanced Robotics, Vol. 17, No. 1, pp. 21–39 (2003)

Also available online - www.vsppub.com

Full paper

A reinforcement learning approach to fail-safe design for multiple space robots — cooperation mechanism without communication and negotiation schemes

KEIKI TAKADAMA [1,2,*], SHUICHI MATSUMOTO [3], SHINICHI NAKASUKA [4] and KATSUNORI SHIMOHARA [1]

[1] *ATR Human Information Science Laboratories, 2-2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0288, Japan*
[2] *Tokyo Institute of Technology, 4259 Nagatsuta-cho, Midori-ku, Yokohama 226-8502, Japan*
[3] *NASDA, Tsukuba Space Center, 2-1-1, Sengen, Tsukuba-shi, Ibaragi 305-8505, Japan*
[4] *University of Tokyo, 7-3-1 Bunkyo-ku, Tokyo 113-8656, Japan*

Received 2 August 2001; revised 25 October 2001; accepted 3 December 2001

Abstract—This paper explores a fail-safe design for multiple space robots, which enables robots to complete given tasks even when they can no longer be controlled due to a communication accident or negotiation problem. As the first step towards this goal, we propose new reinforcement learning methods that help robots avoid deadlock situations in addition to improving the degree of task completion without communications via ground stations or negotiations with other robots. Through intensive simulations on a truss construction task, we found that our reinforcement learning methods have great potential to contribute towards fail-safe design for multiple space robots in the above case. Furthermore, the simulations revealed the following detailed implications: (i) the first several planned behaviors must not be reinforced with negative rewards even in deadlock situations in order to derive cooperation among multiple robots, (ii) a certain amount of positive rewards added into negative rewards in deadlock situations contributes to reducing the computational cost of finding behavior plans for task completion, and (iii) an appropriate balance between positive and negative rewards in deadlock situations is indispensable for finding good behavior plans at a small computational cost.

Keywords: Multiagent-based architecture; multiple space robots; fail-safe design; reinforcement learning; cooperation without communication and negotiation; deadlock.

1. INTRODUCTION

To increase the efficiency of task completion or solve tasks that cannot be completed by a single robot (or satellite), several research works have started with a focus on

*To whom correspondence should be addressed. E-mail: keiki@dis.titech.ac.jp

multiple robots [1–3]. These works have mostly been based on *communications* via ground stations as a global explicit control or *negotiations* among robots, and they have contributed to guaranteeing rational behaviors by robots and supporting cooperation among robots. However, these approaches have the following essential drawbacks in consideration of space applications: (i) the amount of communications is limited by the weights or electrical power levels of the robots, (ii) delays caused by the high communication cost prevent the robots from cooperating in real time or safely, (iii) the communication traffic increases as the number of robots increases, (iv) the robots in a centralized control system are directly affected by system malfunctions and (v) all information cannot always be transmitted correctly due to several types of noise. Even if we suppose that these drawbacks can be solved by future research, robots can actually suffer communication or negotiation failure at any time. In such a case, the communication- or negotiation-based approach will no longer be useful. Furthermore, it is mostly impossible to repair such problems quickly in space. From these facts, it is important to explore mechanisms that do not require communication or negotiation among multiple space robots as one potential approach to tackle the above problems.

However, robots of any type without a communication or negotiation scheme can easily get into a deadlock situation. This is because the robots are unable to acquire accurate global information, only local information. For example, this situation can be easily understood by considering the case where two robots push the same box from the left and right sides, respectively. In this case, robots get into the deadlock situation because they cannot move due to a push from the opposite side without any useful (global) information. In particular, the local information which can be acquired by infrared light sensors or vision mechanisms of robots corresponds to the position of the box, while the global information which is only poorly acquired without a communication or negotiation scheme corresponds to the position of the other robots. As shown in this example, it is difficult for the robots to avoid several types of collisions involving other robots due to inconsistent information. Such situations become more serious particularly in the following cases: (i) the number of robots is increased, (ii) a higher level of cooperation is required among the robots to complete given tasks or (iii) unexpected situations occur. In spite of these deadlock situations, however, most research works have focused on improving the level of task completion in normal situations [4, 5], i.e. cases without deadlock situations. Here, considering the fact that tasks can be completed by coping with deadlock situations in the multiple-robot domain, we should consider a deadlock situation in addition to improving the level of task completion. Based on this notion, this paper aims at exploring a cooperation mechanism that enables multiple space robots to tackle deadlock situations in addition to improving the level of task completion without communications via ground stations or negotiations with other robots. In particular, the cooperation mechanism focused on in this paper is based on the software viewpoint (i.e. the machine learning perspective described in the

next section), not on the hardware viewpoint (i.e. the mechanical or structural perspective).

This paper is organized as follows. The next section starts by briefly explaining our multiagent-based architecture, which does not require communications or negotiations, and introduces our new mechanisms for deadlock situations in addition to improving the level of task completion. A task for multiple space robots is given in Section 3. Section 4 presents our simulations and results, and Section 5 discusses the applicability of our new mechanisms in space applications. Finally, our conclusions are given in Section 6.

2. MULTIAGENT-BASED ARCHITECTURE

Before tackling deadlock situations without communications via ground stations or negotiations with other robots, this section starts by providing a brief explanation of our multiagent-based architecture [6].

2.1. Outline

Our multiagent-based architecture is implemented to enable robots to cooperate without communications or negotiations in environments that include imprecision, uncertainties and partial truths. As shown in Fig. 1, this architecture is composed of many agents, which correspond to robots, and each agent has the same structure comprising the following components:

Interface

- *Detector* and *Effector* translate a part of an environment state into an internal state of an agent and derive actions based on this internal state [7], respectively. In particular, each agent in our architecture can recognize its own environmental

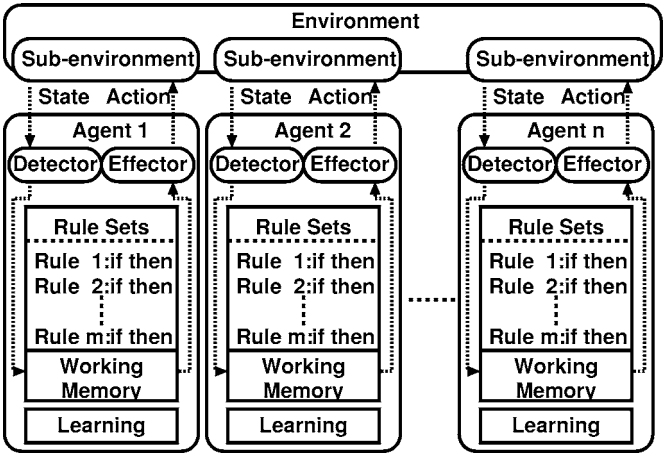

Figure 1. Multiagent-based architecture.

state but not the state of the total environment. This premise reflects the situation that each agent can only acquire local information, but not accurate global information.

Memory

- *Rule sets* are composed of many *if–then* rules with a strength factor (i.e. the worth of rules). Although the agents in this architecture can independently store different *if–then* rules and the sizes of the rules can differ among the agents, the agents in this stage of research store the same numbers of the same rules. (A case with different numbers of different rules is simulated in Ref. [16].) During rule selection, one of these rules is fired according to a *roulette selection mechanism* [8] that probabilistically selects one rule from among plural rules that match a particular environment. More specifically, one rule is selected according to the size of the strength value attached to each rule. Since one action is included in the *then* part in this architecture, one action is performed in each rule selection. Furthermore, all strength values of the rules are numbers from 0 to 1 starting at the same initial value (0.5) and they are varied through the learning mechanisms.
- *Working memory* stores a sub-environmental state that becomes a trigger for firing rules and also stores the internal states of the actions derived from the fired rules.

Mechanism

- *Learning mechanism* changes the strength values of the rules to derive cooperation among the robots without communications or negotiations. Using this learning mechanism, the agents self-generate behavior plans by chaining their own *if–then* rules and self-modify their own plans to cope with the complete given tasks effectively. The *behavior plans*, here, are defined as *sequences of fired rules*, and they are generated and modified by changing the strength values of the rules through a reinforcement learning mechanism (the details are described in the next section). Based on this definition, we use the word *action* for a primitive motion that includes the *then* part of a rule in our architecture and use the word *behavior* for a set of actions such as a sequence of actions in the above plans.

2.2. Learning mechanism

To tackle deadlock situations without communications and negotiations, one of the easiest methods is to prepare behavior plans for deadlock situations in advance. However, it is impossible to prepare perfect plans that cover all possible situations. Furthermore, communication- or negotiation-based robots cannot be counted on to perform according to behavior plans when one or more robots meets with communication or negotiation problems. All of this indicates a limitation in preparing plans in advance for practical and engineering use. To overcome this problem, we believe that robots have to be designed to learn by themselves to acquire their behavior plans and tackle current problems. Towards this goal, our architecture employs *reinforcement learning (RL)* [9] as one of major machine learning approaches to en-

able robots to tackle deadlock situations by self-generating and self-modifying their behavior plans.

2.2.1. Task completion situation. Among the several approaches of reinforcement learning, such as *bucket brigade* [10] or *Q-learning* [11], our architecture employs a *profit sharing* method [12] because of the following reasons: (i) bucket brigade has a tendency to reinforce ineffective rules even if rewards are not provided [13, 14], and (ii) the space state in Q-learning grows exponentially according to the number of agents and profit sharing outperforms Q-learning in POMDP (Partially Observable Markov Decision Process) problems, which are one of essential problems in multiagent environments [15].

As a concrete mechanism for task completion situations, profit sharing reinforces the sequences of all fired rules when completing given tasks. Precisely, the strength values of all of the fired rules are changed as shown in Fig. 2 when completing given tasks. This indicates that the same positive rewards are distributed for all of the fired rules by varying the strength values according to (1). This reinforcement approach is based on the notion that most of all rules contribute to completing given tasks (other reward distributed methods are investigated in Ref. [16]). In Fig. 2, the vertical and horizontal axes indicate the size of the reward and the fired order of rules, respectively. Note that the fired order of rules in the horizontal axis starts at 1 not 0. Furthermore, $ST(i)$, n and $R(> 0)$ in the following equation indicate the strength value of the ith fired rule, the total number of fired rules when completing the given task and the size of the reward, respectively. A small i indicates the first several fired rules.

- RL for task completion in Fig. 2

$$ST(i) = ST(i) + R, \quad \text{where } i = 1, \ldots, n-1, n. \tag{1}$$

2.2.2. Deadlock situation. Since the first version of our architecture proposed in Ref. [6] only employed profit sharing for task completion situations, the robots were prone to get into deadlock situations if rules were reinforced only for task completion. To overcome this problem, this paper proposes the following five new reinforcing methods for deadlock situations and investigates their capabilities in terms of being a fail-safe design for the case where multiple space robots cannot

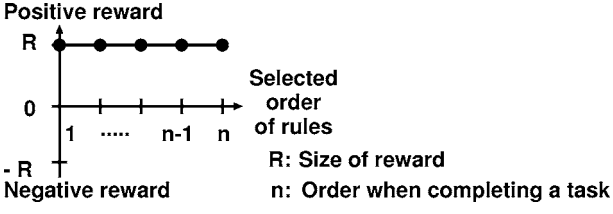

Figure 2. Reinforcement in task completion.

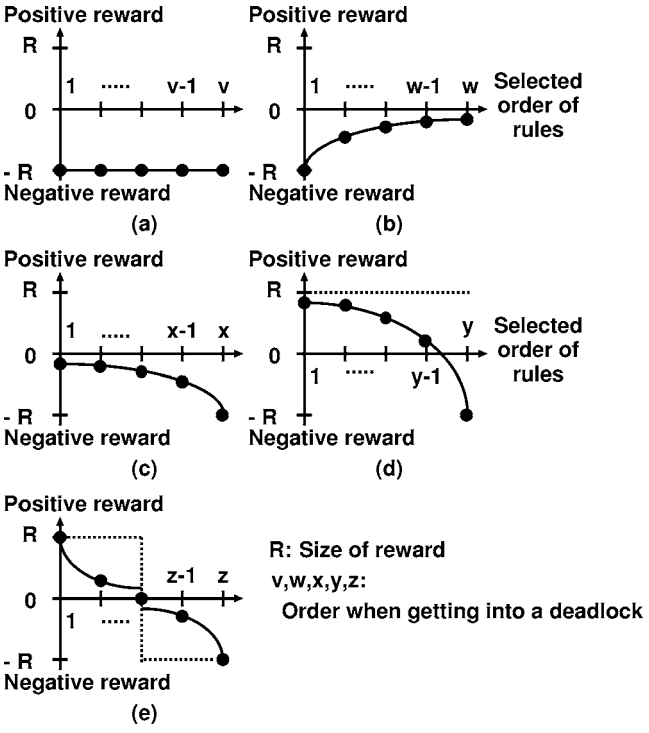

Figure 3. Reinforcement in a deadlock situation.

communicate with ground stations or negotiate with others. Note that the following five methods for deadlock situations are found through a trial and error process of our research, and therefore they do not cover all possible methods.

Figure 3a–e shows the methods targeting deadlock situations. This methods calculate the strength values of all fired rules according to (2)–(6). In Fig. 3, the vertical and horizontal axes have the same definitions in task completion situations. In the following equations, $ST(i)$ and $R(> 0)$ have the same definitions in task completion situations, while v, w, x, y, z and G indicate the total number of fired rules when entering a deadlock situation and the geometric ratio in the range of $0 < G < 1$, respectively.

- RL for deadlock in Fig. 3a

$$ST(i) = ST(i) - R, \quad \text{where } i = 1, \ldots, v-1, v. \tag{2}$$

- RL for deadlock in Fig. 3b

$$ST(i) = ST(i) - R \cdot G^{i-1}, \quad \text{where } i = 1, \ldots, w-1, w. \tag{3}$$

- RL for deadlock in Fig. 3c

$$ST(i) = ST(i) - R \cdot G^{x-i}, \quad \text{where } i = 1, \ldots, x-1, x. \tag{4}$$

- RL for deadlock in Fig. 3d

$$ST(i) = ST(i) + R \cdot (1 - 2 \cdot G^{y-i}), \quad \text{where } i = 1, \ldots, y-1, y. \tag{5}$$

- RL for deadlock in Fig. 3e

$$\begin{aligned} ST(i) &= ST(i) + R \cdot G^{i-1}, \quad \text{where } i = 1, \ldots, z/2 - 1, z/2, \\ ST(i) &= ST(i) - R \cdot G^{z-i}, \quad \text{where } i = z/2 + 1, \ldots, z-1, z. \end{aligned} \tag{6}$$

3. TRUSS CONSTRUCTION TASK

3.1. Problem description

In international space stations (ISS), truss construction is one of the fundamental tasks where beams are combined to create a base of several other works in space. Currently, this job is done by human astronauts. However, it goes without saying that robots should perform such tasks in place of human astronauts not only due to the danger involved, but also to allow the astronauts to carry out higher-level missions.

Figure 4 shows an image of Japanese experimental module (JEM) missions in ISS by multiple space robots. It shows that robots construct a truss through cooperation among them. In particular, HOSV and JEMRMS in this figure indicate multiple space robots currently developed by *NASDA (National Space Development Agency of Japan)* and robot arms of the JEM, respectively. In detail, HOSV stands for

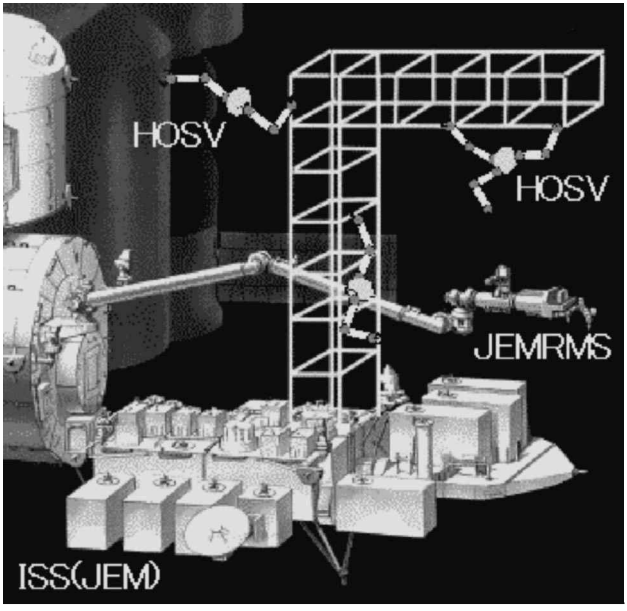

Figure 4. Truss construction task (© NASDA).

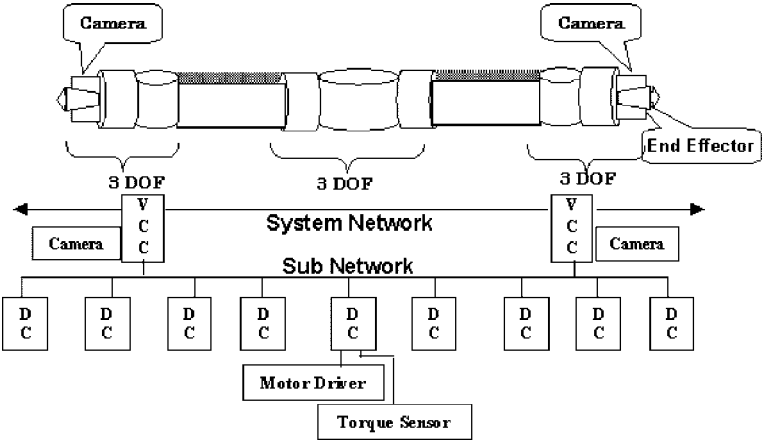

Figure 5. Hyper-OSV (© NASDA).

Hyper-OSV (orbital servicing vehicle) [17] (precisely, the robot shown in this figure is called *RMM (re-configurable Multi-Manipulator)*, and HOSV is a robot that integrates RMM with a space flying system.). The hardware of HOSV is illustrated in Fig. 5. As shown, HOSV has 9 d.o.f. with nine DC joint controllers, two cameras, two end effectors and two VCC concurrent fault tolerant computers. This architecture allows the robot to rotate its positions, move to desired locations, hold a beam and connect multiple beams. The beams are implemented by *information structures* [18] that are connected mechanically and electrically by the end effectors of HOSV.

3.2. Problem setting

Even though the hardware design of HOSV is completed as shown in Fig. 5, it is difficult to directly apply HOSVs to real missions because this task is highly dependent on several mechanical and structural problems or restrictions of HOSVs. From this fact, it is straight-forward to address these issues at the hardware level. However, the method at the software level has the potential not only to make such problems easy or relax restrictions of HOSVs, but also address fail-safe mechanisms which are indispensable when applying HOSVs into real missions. Based on this notion, we focus on the importance of software and start by developing a HOSV simulator to investigate the ability of our proposed methods in terms of fail-safe mechanisms.

In this simulator, the following settings are considered according to the capability and structure of HOSV: (i) HOSV can transport a beam when it is carrying a beam, while HOSV can connect beams when its end effectors are free. This indicates that HOSV cannot perform both actions at the same time, and (ii) HOSV can only obtain *local* environmental information and not *global* environmental information. Examples of local environmental information include beam connecting information, which distinguishes whether a beam is connected or not, information on the

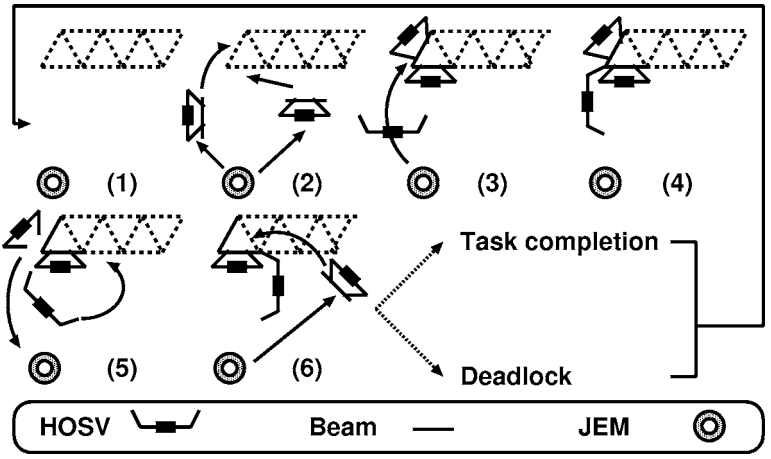

Figure 6. Truss construction.

locations of other nearby HOSVs in order to prevent collision, etc., and (iii) a HOSV has a minimum routine work function such as that the HOSV waits until beams are connected at the beam construction locations or the HOSV connects beams when two beams are brought by other HOSVs at the beam connecting location. Here, note that there are two beam locations in this simulator: a beam *construction* location and a beam *connecting* location. The difference between the two locations is depicted in Fig. 6 and a brief explanation of the difference is described in the next subsection.

3.2.1. How is a truss constructed? Based on the above HOSV settings, HOSVs, for example, construct a truss as shown in Fig. 6, where the box with arms, solid line, double circle and dashed line indicate a HOSV, a beam, the JEM, and a location for truss construction, respectively. Furthermore, a side of the dash-lined triangle depicts a beam construction location while a vertex of the same triangle depicts a beam connecting location. In order to avoid unnecessary behaviors, the HOSVs carry beams and connect them from left to right in Fig. 6. The examples of the three HOSVs shown in Fig. 6 are explained as follows:

(1) The three HOSVs stay at the JEM.

(2) Two of the HOSVs decide to go to the beam construction location holding their own beams at the JEM.

(3) The two HOSVs in (2) arrive at the beam construction location with their own beams. The third HOSV decides to go to the beam connecting location.

(4) The third HOSV in (3) arrives at the beam connecting location and connects the beams.

(5) The HOSV with a beam on the left side in Fig. 6 releases the beam and decides to return to the JEM to bring another beam, while the HOSV connecting the beams decides to go to another connecting location.

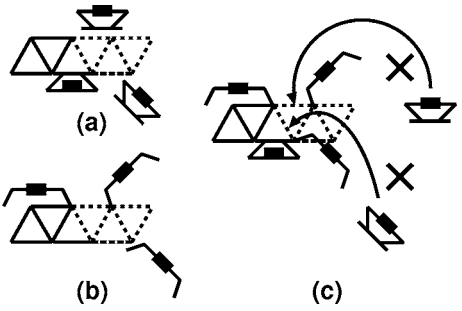

Figure 7. Deadlock situation.

(6) The HOSV that brings another beam goes to the next beam construction location, while the third HOSV arrives at the beam connecting location. By continuing the above beam construction, the HOSVs finally complete the given task or get into a deadlock situation and then goto (1) again.

Concerning deadlock situations, HOSVs get into such situations when (i) all of the HOSVs bring their own beams, (ii) all of the HOSVs go to beam connecting locations, or (iii) some HOSVs are unable to get to beam construction or connecting locations due to narrow spaces between waiting HOSVs as shown in Fig. 7a, b and c, respectively.

3.2.2. What do HOSVs learn? In order to construct a truss effectively through cooperation among multiple HOSVs, all of the HOSVs at the JEM must learn to decide whether they will go to beam construction locations to bring their own beams or go to beam connecting locations to connect beams, while all of the HOSVs at the beam construction and beam connecting locations must also learn to decide whether they will return to the JEM to get other beams or go to other beam connecting locations to connect the next beams. This learning in our architecture is continued to acquire more appropriate behavior plans than the current ones by starting from the same initial location of the JEM after completing given tasks or getting into deadlock situations as shown in Fig. 6. In particular, such appropriate behavior plans are generated and modified by changing the strength values of the rules through the reinforcement learning mechanism described in Section 2.

However, we assume that the HOSVs in this simulation are unable to communicate via ground stations or negotiate with other HOSVs, and, accordingly, the HOSVs have no function to receive rewards from the outside. Therefore, the HOSVs acquire their behavior plans by self-generated rewards. Precisely, *positive* rewards are self-generated when HOSVs can no longer find new beams at the JEM, which means that all of the beams have been brought to the beam construction locations, while *negative* rewards are self-generated when HOSVs stay at the beam construction or beam connecting locations more than a certain period of time, which means deadlock.

3.3. Evaluation criteria

As evaluation criteria in truss construction tasks, we employ the following two indexes:

$$Solution = Steps, \tag{7}$$

$$Computational\ cost = Iterations. \tag{8}$$

The first index (*Solution*) evaluates the steps after HOSV learning. In our simulation, one step is counted when all of the HOSVs perform one action according to the fired rules. This index calculates the number of performed actions of each HOSV. The second index (*Computational cost*), on the other hand, evaluates the iterations counted until HOSV learning. In our simulation, one iteration is counted when the HOSVs construct a truss or get into a deadlock situation. In other words, iterations are trial repeating numbers from the same initial location of the JEM until task completion or a deadlock situation and they are counted until the values of the steps converge. From the above evaluation criteria, the targets of HOSVs are (i) to acquire good behavior plans that construct a truss in a small number of steps (i.e small number of performed actions) and (ii) to acquire such plans within a few iterations.

3.4. Rule set design

The *if–then* rules of the HOSVs are designed for truss construction tasks as follows. An example involving rule sets is shown in Fig. 8.

- The condition part
  - A previously performed action. In detail, the following four types are employed: stay (labeled as 0), go to a beam construction location and bring a beam (labeled as 1), go to a beam connecting location (labeled as 2) or go back to the JEM (labeled as 3)
  - Current location information. In detail, the following four types are employed: staying at the JEM (labeled as 0), at a beam construction location (labeled as 1), at a beam connecting location (labeled as 2) or in space (labeled as 3)
  - Remaining beam information, which distinguishes whether there are beams remaining in the JEM or not (labeled as 1 or 0)
  - Connected beam information which distinguishes whether beams are connected or not (labeled as 1 or 0)
- The action part
  - An action (the four types described above)
- The strength part
  - A strength value

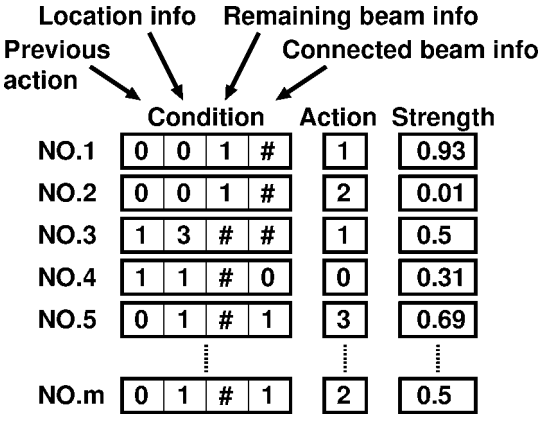

Figure 8. Rule set.

According to this rule design, the rules shown in Fig. 8 are interpreted as follows. Note that the mark # indicates a *don't care*.

- The first and second rules indicate that *if* a previous action is 'stay' and a current location is in the 'JEM' and beams 'remain', *then* go to a 'beam construction location' or go to a 'beam connecting location', respectively. When these rules are matched with the current situation of the HOSV, one of two rules is selected probabilistically according to the size of the strength value attached to each rule (i.e. 0.93 and 0.01) as described in Section 2.1.
- The third rule indicates that *if* a previous action is go to a 'beam construction location' and the current location is in 'space', *then* continue to go to the 'beam construction location'.
- The fourth rule indicates that *if* a previous action is go to a 'beam construction location' and the current location is in the 'beam construction location' and beams are 'not connected', *then* 'stay'.
- The fifth rule indicates that *if* a previous action is 'stay' and the current location is in a 'beam construction location' and beams are 'connected', *then* go back to the 'JEM'.

4. SIMULATION

4.1. Experimental design

To investigate our reinforcement learning methods in terms of fail-safe design for multiple space robots, this paper conducted the following five truss construction simulations. Note that the reinforcement learning method for task completion shown in Fig. 2 was used in all five cases.

- Case (a): Reinforcement in Fig. 2 + Fig. 3a
- Case (b): Reinforcement in Fig. 2 + Fig. 3b

- Case (c): Reinforcement in Fig. 2 + Fig. 3c
- Case (d): Reinforcement in Fig. 2 + Fig. 3d
- Case (e): Reinforcement in Fig. 2 + Fig. 3e

All of the cases were tested with five to 10 HOSVs that constructed a truss by combining 13 beams, and then the results of the above five cases were compared in terms of both the solution (i.e. the steps after HOSV learning) and the computational cost (i.e. the iterations counted until HOSV learning), as described in Section 3.3. Parameters R and G were defined as 1 and 0.5, respectively. We had already confirmed that the tendency in the results would not drastically change according to the parameter settings.

4.2. Simulation results

Figure 9 shows results of the steps required to construct a truss after HOSV learning (Fig. 9i) and the iterations counted until HOSV learning (Fig. 9ii). In Fig. 9, the upper vertical, the lower vertical and the horizontal axes indicate the steps, the iterations and the numbers of HOSVs, respectively. The five boxes for each HOSV indicate the respective results of the cases (a)–(e), and the cross marks in cases (a) and (b) indicate that the HOSVs could not get out of their deadlock situations. Note that both steps and iterations are averaged from five different random seeds.

From Fig. 9, we find the following implications: (i) the methods in Fig. 3a and b do not allow HOSVs to construct a truss as they cannot get out of deadlock situations, while the methods in Fig. 3c–e allow HOSVs to complete the truss

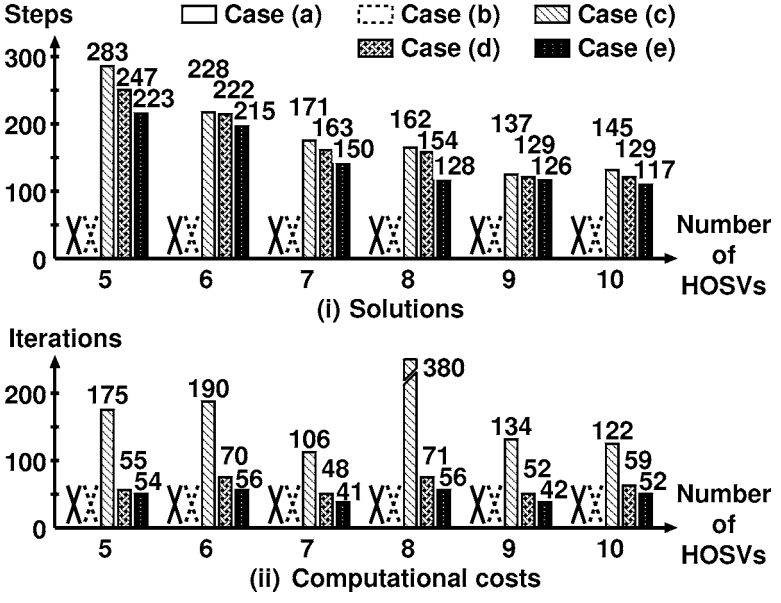

Figure 9. Solutions and computational costs.

construction; (ii) the iterations in Fig. 3d and e are less than half of those in Fig. 3c; and (iii) the method in Fig. 3e finds better solutions at smaller computational costs than the others. In other words, HOSVs using the method in Fig. 3e can acquire good behaviors plans within a few iterations, i.e. construct a truss in a small number of actions.

5. DISCUSSION

5.1. Findings and implications

5.1.1. Importance of behaviors in the first several steps. Figure 9 shows that HOSVs with the reinforcement learning methods given in Fig. 3a and b cannot complete truss construct tasks. The reasons are summarized that the methods for deadlock situations entirely vary the behavior plans of HOSVs and this greatly influences the behavior plans of other HOSVs. It is important to note here that the first parts of the behavior plans in this case are changed by varying the strength values of the rules with strong negative rewards. From this factor, the unstable behavior plan acquisition in the above methods prevents HOSVs from cooperating with each other and, accordingly, HOSVs are unable to get out of deadlock situations.

In contrast to these methods, HOSVs with the reinforcement learning methods given in Fig. 3c–e can complete given tasks. The reasons are summarized as follows: (i) the behavior plans acquired with the above methods become stable to some degree because the first several selected rules are not reinforced with strong negative rewards or they are reinforced with positive rewards even in the case of deadlock situations and (ii) this stable behavior plan acquisition increases the chances of getting out of deadlock situations through cooperation based on other stable behavior plans. In particular, the behavior plans in Fig. 3c–e do not entirely change because the methods in Fig. 3c–e mainly vary the last several selected rules.

From the above analysis, the first conclusion of this paper is that the behaviors in the first several steps are important for multiple robots to get out of deadlock situations.

5.1.2. Importance of positive rewards added into negative rewards. Figure 9 shows that HOSVs with the reinforcement learning methods given in Fig. 3d and e reduce the iterations to less than half of the iterations in Fig. 3c. To analyze this result, we start by focusing on the methods in Fig. 3d and e, which distribute positive rewards in the first several steps even in deadlock situations, although the general tendency is to distribute negative rewards in such cases. This consideration shows that the behavior plans of HOSVs are established in the first few several iterations because the first several selected behaviors in Fig. 3d and e are reinforced with large positive rewards. This quick behavior plan establishment clearly reduces the

iterations as shown in Fig. 3d and e. Furthermore, another effect of the methods in Fig. 3d and e is that behaviors performed before getting into deadlock situations do not always lead to deadlock situations, but some of them, especially in the first several steps, contribute to completing the tasks by having HOSVs cooperate.

From the above analysis, the second conclusion of this paper is that a certain amount of positive rewards added into negative rewards for deadlock situations can contribute to reducing the iterations.

5.1.3. Appropriate balance between positive and negative rewards. In a comparison of the results between the reinforcement learning methods in Fig. 3d and e, Fig. 9 shows that HOSVs using the method in Fig. 3e can complete a given task in fewer steps and iterations than HOSVs using the method in Fig. 3d. This is because the rules in Fig. 3e are equally reinforced with both positive and negative rewards, while almost all of the rules in Fig. 3d are reinforced with positive rewards even in deadlock situations. As a result of an appropriate balance between positive and negative rewards, the method in Fig. 3e finds good behavior plans that complete given tasks with a few steps by *exploring* the search space and reduces the iterations by *exploiting* a part of the previously acquired behavior plans. This indicates that the method in Fig. 3e can cope with the trade-off relationship between exploration and exploitation by the appropriate balance of rewards.

From the above analysis, the third conclusion of this paper is that an appropriate balance between positive and negative rewards is indispensable for finding good solutions (i.e. good behavior plans that complete given tasks in a few steps) at small computational costs (i.e. a few iterations). This indicates that this reinforcement approach is effective for practical and engineering use even when robots cannot communicate via ground stations or negotiate with others. In particular, the acquisition of good behavior plans at a few iterations can help robots reduce both the time and cost necessary for robot learning, which is indispensable in the limited time of space missions.

5.2. Towards fail-safe design for multiple space robots

5.2.1. Key points for fail-safe design. From the above discussion, the following findings and implications are revealed through an analysis of simulation results: (i) the first several planned behaviors are important for multiple robots to get out of deadlock situations, (ii) a certain amount of positive rewards added into negative rewards in deadlock situations contributes to reducing the computational cost of finding behavior plans for task completion, and (iii) an appropriate balance between positive and negative rewards in deadlock situations is indispensable for finding good behavior plans at a small computational cost.

From these findings and implications, what kinds of contributions can we make towards fail-safe design for multiple space robots? This section explores this answer.

First, point (i) suggests that we pay careful attention to evaluating the first several planned behaviors. This is because such behaviors have the potential to determine the method of cooperation among robots. Second, point (ii) suggests that we do not evaluate behaviors according to one criterion but to multiple criteria in order to improve the level of task completion. This is because behaviors performed before getting into deadlock situations do not always lead to deadlock situations — some of them contribute to completing the tasks. Finally, point (iii) suggests that we do not evaluate behaviors with any bias but evaluate them impartially in order to find good behavior plans at a small computational cost. In this case, a sequence of behaviors is also necessary for keeping an impartial evaluation because an evaluation becomes biased when the number of behaviors is small. From the above discussion, we can conclude that the following three suggestions provide potential approaches towards fail-safe design for multiple space robots: (i) control of the first several planned behaviors, (ii) employing multiple evaluation criteria and (iii) impartial evaluation of a sequence of behaviors.

5.2.2. Validity of our methods. This paper showed the effectiveness of our methods in terms of getting out of deadlock situations in addition to improving the level of task completion. However, the following important issues still remain: (i) a comparison with other methods and (ii) physical experiments. Although further simulations and experiments in future works are necessary to address the above two issues directly and completely, the following discussion can be made at the current stage of our research.

For the comparison issue, our previous research addressed the same example in simulations and showed that our methods which can only recognize local information outperformed the method which can acquire accurate global information through communications via ground stations [19]. This result increases the validity of our methods. The physical experiment issue, we have not yet addressed it directly. However, our previous research addressed the case where one or more robots failed or became inoperative in the same example and showed the effectives of our former methods in simulations [6, 20]. This indicates that our methods work well not only in stable and ideal environments, but also in unstable and real-life environments. Although our previous research was limited to bridging a gap between computer simulations and physical experiments, the result also increases the validity of our methods.

6. CONCLUSION

This paper explored a fail-safe design for multiple space robots, which enables robots to complete given tasks even when they can no longer be controlled due to a communication accident or negotiation problem. As the first step towards this goal, we proposed new reinforcement learning methods that can help robots get

out of deadlock situations in addition to improving the degree of task completion without communications via ground stations or negotiations with other robots, and investigated the capabilities of these methods in truss construction tasks considering future HOSV missions. Based on the simulation results, this paper arrived at the conclusion that our new reinforcement learning methods have great potential to contribute towards fail-safe design for multiple space robots without communications and negotiations. In detail, the following three points provide potential approaches towards such design: (i) control of the first several planned behaviors, (ii) employing multiple evaluation criteria and (iii) impartial evaluation to a sequence of behaviors.

Although this paper only conducted simulation experiments and the truss construction tasks did not cover entire space application tasks, the following implications were revealed. (i) Although we tend to reinforce with negative rewards when robots get into deadlock situations, the first several planned behaviors in the multiple robot domain must not be reinforced with negative rewards even in deadlock situations. This means that the first several planned behaviors are very important for getting out of deadlock situations in multiple-robot environments. (ii) A certain amount of positive rewards added into negative rewards in deadlock situations contributes to reducing the computational cost of finding behavior plans for task completion. This indicates that positive rewards are required even in deadlock situations to derive cooperation among multiple robots. (iii) An appropriate balance between positive and negative rewards in deadlock situations is indispensable for finding good solutions (i.e. good behavior plans that complete given tasks within a few steps) at small computational costs (i.e. small iterations). In this sense, the reinforcement learning in Fig. 3e contributes to reducing both the time and cost for robot learning, which is indispensable in the limited time of space missions. Finally, the implications obtained are based on a multiagent architecture and therefore they can apply not only to multiple space robots, but also to a lot of other multiagent-type applications such as multiple satellites.

Future research includes:

- Improvement of the reinforcement learning methods for task completion in addition to considering the proposed reinforcement learning methods for deadlock situations.
- Theoretical analysis of our reinforcement learning methods for multiple robots.
- Further comparisons of our reinforcement learning methods with other methods.
- Investigation of our reinforcement learning methods with physical HOSV robots.
- Investigation on the capabilities of our reinforcement learning methods when the number of robots is increased in excess of the numbers in this paper.

REFERENCES

1. S. Yuta and S. Premvuti, Coordinating autonomous and centralized decision making to active cooperative behaviors between multiple mobile robots, in: *Proc. IEEE Int. Workshop on Intelligent Robots and Systems*, pp. 1566–1575 (1992).
2. K. Asama, H. Ozaki, A. Itakura, A. Matsumoto, Y. Ishida and I. Endo, Collision avoidance among multiple mobile robots based on rules and communication, in: *Proc. IEEE Int. Workshop on Intelligent Robots and Systems*, pp. 1215–1220 (1991).
3. L. E. Parker, ALLIANCE: an architecture for fault tolerant, cooperative control of heterogeneous mobile robots, in: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 776–783 (1994).
4. M. J. Mataric, Reward functions for accelerated learning, in: *Proc. 11th Int. Conf. on Machine Learning*, pp. 181–189 (1994).
5. S. Sen, M. Sekaran and J. Hole, Learning to coordinate without sharing information, in: *Proc. 11th Natl. Conf. on Artificial Intelligence*, pp. 426–431 (1991).
6. K. Takadama, H. Kasahara, L. Huang, M. Watabe, H. Ii, K. Shimohara and S. Nakasuka, Organizational learning agents for task scheduling in space crew and robot operations, in: *Proc. 5th Int. Symp. on Artificial Intelligence, Robotics and Automation in Space*, pp. 561–568 (1999).
7. S. J. Russell and P. Norving, *Artificial Intelligence: A Modern Approach.* Prentice-Hall, Englewood Cliffs, NJ (1995).
8. D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, Reading, MA (1989).
9. R. S. Sutton and A. G. Bart, *Reinforcement Learning — An Introduction.* MIT Press, Cambridge, MA (1998).
10. J. H. Holland, Properties of the bucket brigade algorithm, in: *Proc. 1st Int. Conf. on Genetic Algorithms*, pp. 1–7 (1985).
11. C. J. C. H. Watkins and P. Dayan, Technical note: Q-learning, *Machine Learning* **8**, 55–68 (1992).
12. J. J. Grefenstette, Credit assignment in rule discovery systems based on genetic algorithms, *Machine Learning* **3**, 225–245 (1988).
13. R. L. Riolo, Bucket brigade performance: long sequence of classifiers, in: *Proc. 2nd Int. Conf. on Genetic Algorithms*, pp. 184–195 (1987).
14. R. E. Smith and D. E. Goldberg, Variable default hierarchy separation in a classifier system, in: *Foundation of Genetic Algorithms*, pp. 148–167. Morgan Kaufman, San Mateo, CA (1991).
15. S. Arai and K. Sycara, Effective learning approach for planning and scheduling in multi-agent domain, in: *Proc. 6th Int. Conf. of the Society for Adaptive Behavior*, pp. 507–516 (2000).
16. K. Takadama, S. Nakasuka and K. Shimohara, Robustness in organizational-learning oriented classifier system, *J. Soft Comput.* **6** (3-4), 229–239 (2002).
17. S. Matsumoto, Y. Wakabayashi and Y. Watanabe, System design of the Hyper-OSV (Orbital Servicing Vehicle), in: *Proc. 10th Workshop on Astrodynamics and Flight Mechanics* (2000).
18. H. Ueno, Y. Wakabayashi and S. Matsumoto, Reconfigurable on-orbit experimental testbed by using brachiate manipulators, in: *Proc. 44th Conf. on Space Science and Technology* (2000).
19. K. Takadama, K. Hajiri, T. Nomura, M. Okada, S. Nakasuka and K. Shimohara, Learning model for adaptive behaviors as an organized group of swarm robots, *Int. J. Artificial Life Robotics* **2** (3), 123–128 (1998).
20. H. Kasahara, K. Takadama, S. Nakasuka and K. Shimohara, A troubleshooting mechanism based on an organizational learning model for multiple robots, in: *Proc. 1998 Int. Technical Conf. on Circuits Systems, Computer and Communications*, pp. 1023–1026 (1998).

ABOUT THE AUTHORS

Keiki Takadama received his ME degree from Kyoto University in 1995 and his DEng degree from the University of Tokyo in 1998, respectively. He joined the Human Information Science Laboratories of the Advanced Telecommunications Research Institute (ATR) International, and since 2002 has been working for Tokyo Institute of Technology as a Lecture. His research interests include multi-agent learning, distributed agent design, autonomous and intelligent system, and reinforcement learning. He is a member of IEEE, ISAB, and other major AI- and robotics-related academic societies.

Shuichi Matsumoto received his BE degree from the University of Tokyo in 1989 and his MS degree in Aeronautics and Astronautics from Stanford University in 1999. He is the Research Engineer of the Expert Group for Guidance, Control and Dynamics at NASDA, where he has been working on guidance and control systems for future space applications for 10 years; he has been doing research on space robotics since 1999. His research interests are guidance, navigation and control for space vehicles including space robots. He is currently a Visiting Researcher at the Department of Mechanical Engineering at MIT. He is a member of AIAA, ION, SICE and JSASS.

Shinichi Nakasuka received his BE, ME and DEng degrees from from University of Tokyo in 1983, 1985 and 1988, respectively. He joined IBM Research, Tokyo Research Laboratory from 1988 to 1990, and has been working for the University of Tokyo since 1990, as a Lecturer and an Associate Professor. He joined the Department of Computer Science of the University of Maryland as a Visiting Researcher in 1996. His research interests include autonomous and intelligent systems, multi-robot architecture, machine learning, and their applications to space technology. He is a member of AIAA, SICE, JSASS and the Japan Rocket Society.

Katsunori Shimohara received his BE and ME degrees in Computer Science and Communication Engineering from Kyushu University in 1976 and 1978, respectively, and his DEng degree from Kyushu University in 2000. He is Director of the Human Information Science Laboratories of the Advanced Telecommunications Research Institute (ATR) International, and Guest Professor in the Graduate School of Informatics, Kyoto University, both in Japan. His research interests include human communication mechanisms, evolutionary systems, the artificial brain and emotion, human–system interactions, and genome informatics. He is a member of IEEE, IEICE, JNNS, JSAI, HIS and the Japan Virtual Reality Society.