ORGANIZATIONAL LEARNING AGENTS FOR TASK SCHEDULING IN SPACE CREW AND ROBOT OPERATIONS

○ **Keiki Takadama**
ATR Human Information Processing Research Labs.
2-2 Hikaridai, Seika-cho
Soraku-gun, Kyoto
619-0288 Japan
keiki@hip.atr.co.jp
Tel: +81-774-95-1007
Fax: +81-774-95-1008

Hitomi Kasahara
Nara Institute of Science and Technology
8916-5 Takayama-cho
Ikoma City, Nara
630-0101 Japan
hitomi-k@is.aist-nara.ac.jp
Tel: +81-743-72-5256
Fax: +81-743-72-5259

Linchun Huang
Japan Advanced Institute of Science and Technology
1-1, Asahidai
Tatsunokuchi, Ishikawa
923-1292 Japan
huanglc@jaist.ac.jp
Tel: +81-761-51-1699
Fax: +81-761-51-1116

Masakazu Watabe
ATR Human Information Processing Research Labs.
2-2 Hikaridai, Seika-cho
Soraku-gun, Kyoto
619-0288 Japan
xmwatabe@hip.atr.co.jp
Tel: +81-774-95-2665
Fax: +81-774-95-1008

Hiromitsu Ii
Univ. of Tokyo
7-3-1 Bunkyo-ku
Tokyo 113-8656 Japan
hiromi@space.t.u-tokyo.ac.jp
Tel: +81-3-3481-4486
Fax: +81-3-3481-4585

Katsunori Shimohara
ATR Human Information Processing Research Labs.
2-2 Hikaridai, Seika-cho, Soraku-gun
Kyoto 619-0288 Japan
katsu@hip.atr.co.jp
Tel: +81-774-95-1070
Fax: +81-774-95-1008

Shinichi Nakasuka
Univ. of Tokyo
7-3-1 Bunkyo-ku
Tokyo 113-8656 Japan
nakasuka@space.t.u-tokyo.ac.jp
Tel: +81-3-3481-4452
Fax: +81-3-3481-4585

Abstract

This paper explores rescheduling and reorganization abilities of our organizational learning model in the following two important applications in space: crew task scheduling in a space shuttle/station and task planning for truss construction with multiple space robots. Through intensive simulations of the above two tasks, the following experimental results have been obtained: (1) Our model provides good feasible schedules quickly in the case of rescheduling, and it keeps the computational cost for rescheduling low; (2) Plans generated by our model keep or recover efficiency in tasks when robots are added, removed, or exchanged among robot groups; and (3) The integration of (a) learning mechanisms, (b) rule based systems with evolutionary approaches, and (c) multiagent approaches is effective in rescheduling/replanning problems.

Keywords: crew task scheduling, planning for multiple space robots, multiagent system, organizational learning, learning classifier system

1 Introduction

In space tasks, unexpected situations often occur that avoid experiments or works from going according to schedules or plans. For example, the crew task schedule on a space shuttle/station, a type of job-shop scheduling problem, is often modified due to instrument/crew anomalies, mission changes, or other schedule change requirements. As another example, pre-determined plans for multiple space robots fail to make sense when one or more robots become failed or inoperative. In the above two cases, new acceptable schedules or plans, even if not optimum, must be obtained as quickly as possible to minimize the time loss. Thus, it is hard in this case to employ conventional methods based on operations research, expert systems, domain-specific heuristic algorithms, or meta-heuristics methods [Osman 96] such as *genetic algorithms* (GAs) [Goldberg 89] or *simulated annealing* (SA) [Aarts 89] for practical and engineering use. This is because (1) the above methods require a lot of time or high computational costs even for small modifications, (2) the methods are difficult to cover all unexpected situations, and (3) even small modifications affect whole systems.

To overcome these problems, recent research on (1) learning mechanisms, (2) rule based systems with evolutionary approaches, and (3) multiagent approaches has studied new possibilities in scheduling or planning domains. For instance, Zhang showed that a reinforcement learning approach found a good feasible schedule more quickly than Zweben's method which is based on simulated annealing [Zweben 94] in the NASA space shuttle payload processing task [Zhang 95]. Since this method can utilize results acquired through the learning, times for making a schedule or a computation costs are reduced. Tamaki showed the generality/applicability of production systems with an evolutionary approach in the case of environmental changes [Tamaki 99], which indicates the potential to cover some unexpected situations. Furthermore, Fujita and Iima

Proc. Fifth International Symposium on Artificial Intelligence, Robotics and Automation in Space, 1-3 June 1999 (ESA SP-440)

showed multiagent approaches contribute to finding good schedule in a reasonable time in rescheduling problems [Fujita 96, Iima 99].

However, research in these three areas seems to have concentrated on improvements in particular methods or techniques independently, in spite of the fact that these components complement each other. Therefore, this paper employs our model that integrates the above three components from multi-strategic standpoints [Takadama 98a, Takadama 99a] and explores this model's possibility in rescheduling and re-planning problems.

This paper is organized as follows. Section 2 starts by explaining our model, and Section 3 describes two space tasks for scheduling and planning. Section 4 presents our simulations, and the possibilities of our model is discussed in Section 5. Finally, our conclusions are given in Section 6.

2 Organizational-learning oriented Classifier System

Our Organizational-learning oriented Classifier System (OCS) [Takadama 98a, Takadama 99a] is a GBML (Genetics-Based Machine Learning) architecture. OCS is composed of many Learning Classifier Systems (LCSs) [Goldberg 89, Holland 78], which are extended to introduce the concepts of organizational learning (OL) [†] studied in organization and management science [Argyris 78, March 91, Cohen 95]. Since LCS is equipped with (1) an environmental adaptation function via reinforcement learning mechanisms, (2-a) a problem solving function via rule-based production systems, and (2-b) rule generation/exchange mechanisms via genetic algorithms, and (3) OCS is an extension of LCSs to multiagent environments, it is easily found that OCS includes (1) learning mechanisms, (2) rule based systems with evolutionary approaches, and (3) the multiagent approaches mentioned in the previous section.

2.1 Aim of agent and function

In OCS, agents (jobs of crews or robots in this paper) are implemented by their own LCSs, and they divide given problems by acquiring their own appropriate *functions* through interaction among agents in order to solve problems that cannot be solved at an individual level. Based on this way of problem solving, the *aim* of the agents is defined as finding appropriate *functions*. Furthermore, these functions are acquired through the change of agents' rule sets (*i.e.*, rule base), and thus a *function* is defined as a rule set. In particular, a rule set drives a certain sequence of actions such as $ABCBC\cdots$, in which the A, B and C actions are primitive actions.

Note that the learning needed to acquire appropriate functions in some agents is affected by the function acquisition of other agents. For example, some agents are affected when one of the A, B, or C actions of other agents changes through learning or when the fired order of the A, B, and C actions of other agents changes.

2.2 Architecture

As shown in Fig. 1, OCS is composed of many agents, and each agent has the same architecture, which includes the following problem solver, memory, and mechanisms. In this model, each agent can recognize its own environmental state but cannot recognize the state of the total environment. Note that the component concerning organizational knowledge is not used in this experiment because it is a different component as compared with the three components mentioned in section 1 and because the aim of this paper is to explore the possibility of the integration of these three components.

< **Problem Solver** >

- **Detector and Effector** change a part of an environmental state into an internal state and change an internal state into an action [Russell 95], respectively.

< **Memory** >

- **Individual knowledge memory** stores a rule set (a set of CFs (classifiers)) as individual knowledge. In OCS, agents independently store different CFs that are composed of *if-then* rules that have a strength factor (*i.e.*, the worth of rules). In particular, one primitive action is included in the *then* part.
- **Working memory** stores the recognition results of sub-environmental states and also stores the internal state of an action of fired rules.
- **Rule sequence memory** stores a sequence of fired rules in order to evaluate them. This memory is cleared after the evaluation.

< **Mechanisms** >

- **Roulette selection** probabilistically selects one rule from among plural rules that match a particular environment. In detail, one rule is selected according to the size of the strength attached to each rule. Since each rule includes one primitive action, one action is performed in each roulette selection.
- **Reinforcement learning, rule generation, rule exchange, and organizational knowledge reuse mechanisms** are reinterpreted from the four kinds of learning in OL (Details

[†]Detailed introduction to the concepts of OL is discussed in [Takadama 99a].

are described later except for the organizational knowledge reuse mechanism).

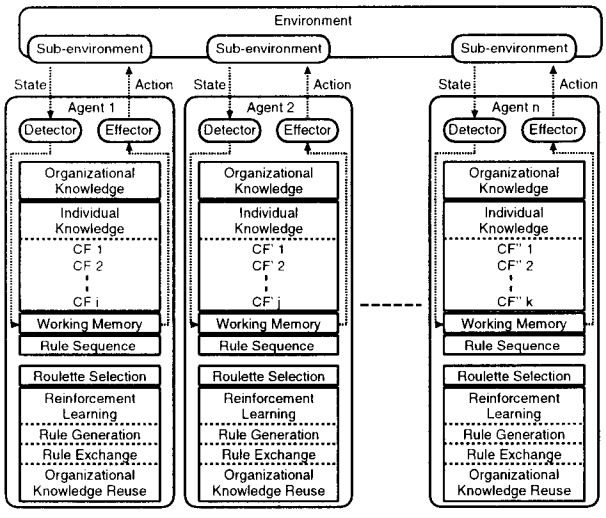


Figure 1: **OCS Architecture**

2.3 Learning in OCS

2.3.1 Reinforcement learning mechanism

In OCS, the reinforcement learning (RL) mechanism enables agents to acquire their own appropriate actions that are required to solve given problems. In particular, RL supports agents to learn the appropriate order of the fired rules by changing the strength of the rules. In detail, OCS employs a *profit sharing* method [Grefenstette 88], which reinforces a sequence of rules at once when agents obtain some rewards [†].

2.3.2 Rule generation mechanism

The rule generation mechanism in OCS creates new rules when none of the stored rules match the current environmental state. In particular, when the number of rules is `MAX_CF` (maximum number of rules), the rule with the lowest strength is removed and a new rule is generated. In a process of rule generation, the condition (if) part of a rule is created to reflect the current situation, the action (then) part is determined at random, and the strength value of the rule is set to the initial value. Furthermore, if the situation does not change because the same rules are repeatedly selected, the strength of the rules is temporarily decreased and these rules become candidates that may be replaced by new rules.

2.3.3 Rule exchange mechanism

In OCS, agents exchange rules with other agents at a particular time interval (`CROSSOVER_STEP`[‡]) in order to solve given problems that cannot be solved at an individual level. In this mechanism, a particular number ((the number of rules)×`GENERATION_GAP`[†]) of rules with low strength values are replaced by rules with high strength values between two arbitrary agents. For example, when agents X and Y are selected as shown in Fig. 2, the CFs in each agent are sorted by order of their strength (upper CFs have high strength values), and $CF_{j-2} \sim CF_j$ and $CF'_{k-2} \sim CF'_k$ in this case are replaced by $CF'_1 \sim CF'_3$ and $CF_1 \sim CF_3$, respectively. However, rules that have strength higher than a particular value (`BORDER_ST`) are not replaced to avoid unnecessary crossover operations. The strength of replaced rules are reset to their initial values. This is because effective rules in some agents are not always effective for other agents in multiagent environments.

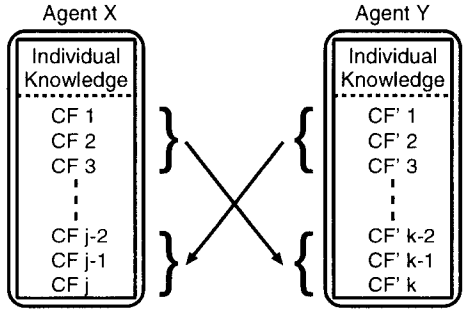


Figure 2: **Rule exchange mechanism**

2.4 Supplemental Setup

In addition to the above mechanisms, OCS is set up as follows: In the beginning, a particular number (`FIRST_CF`) of rules in each agent is generated at random, and the strength values of all rules are set to the same initial value.

3 Task Domain

3.1 Crew Task Scheduling

3.1.1 Problem Description

In the crew task scheduling of a space shuttle/station, many crew jobs must be scheduled under hard resource constraints. In particular, jobs in this task are components of missions, and they should be assigned while satisfying the following constraints.

1. **Power of space shuttle/station:** Each job needs a particular size of power (from 0% to 100%) in experiments, but the summation of the power of all jobs at each time must not be more than 100%.
2. **Link to the ground station:** Some jobs need to use a link in experiments, but only one job can use it at each time. Due to the orbit of

[†]The detail credit assignment in OCS was proposed in [Takadama 98b].

[‡]This step is defined in section 3.1.2 and 3.2.2.

[†]The ratio of operated rules.

the spacecraft, none of the jobs can use the link during a certain time.

3. **Machine A:** Some jobs need to use a machine A in experiments, but only one job can use it at each time. Examples of such machines involve computers, voice recorders, and so on.
4. **Machine B:** The condition is the same as for machine A.
5. **Priority order in jobs:** In a mission unit, jobs have their priority orders (from 1 to the total number of jobs where a smaller number means a higher priority). Jobs in a mission must be scheduled to satisfy their priority orders.
6. **Crew assignment types:** The crew is divided into the following two types: Mission Specialist (MS) and Payload Specialist (PS). The former is mainly in charge of experiments, and the latter supports experiments. In a specific assignment, "the required number of crew members," "the necessary persons," and "the necessary crew assignment types" are decided for each job. For the third element, one of the following crew assignment types must be satisfied: (a) Anybody, (b) PS only (PS is not specified), (c) One specified PS with somebody, (d) One specified MS with somebody, and (e) Combination of PS and MS (PS and MS are not specified). These types are based on the space shuttle missions.

3.1.2 Problem Setting

In this task, each job is designed as an agent in OCS, and each learns to acquire an appropriate sequence of actions that minimizes the total scheduling time. In detail, jobs have 15 primitive actions such as movements for satisfying power constraints, or movements toward an earlier time in a schedule if all constraints are satisfied. Furthermore, jobs can only recognize the situations of their neighbors.

As the concrete problem setting without anomalies, all jobs are initially placed at random without considering overlaps and the six constraints described in the previous section, and therefore a schedule at this time is not feasible. After this initial placement, the jobs start to perform some primitive actions in order to reduce the overlap or to satisfy the constraints while minimizing the total scheduling time. When the value of the total time converges with a feasible schedule, all jobs evaluate their own sequences of actions according to the value of the total time. Then, the jobs restart from the initial placement to acquire more appropriate sequences of actions which find shorter times. In this cycle, one *step* is counted when all jobs perform one primitive action, and one *iteration* is counted when the value of the total time converges with a feasible schedule.

In the case of anomalies, on the other hand, there are two ways of scheduling in OCS: (1) the same way as a case without anomalies (reschedule from the beginning) and (2) all jobs start from the placement of a current schedule that satisfies all constraints except for the anomaly parts (reschedule from the current schedule). Especially in the latter case, only jobs that do not satisfy constraints due to anomalies change their locations in the schedule, and thus a modified schedule can be obtained quickly.

3.1.3 Index of Evaluation

In this task, the following two indexes are evaluated:

- Goodness = *total scheduling time*.
- Computational cost
$$= \sum_{i=start}^{iteration\_in\_convergence} step\ (i)$$

The first index (*goodness*) evaluates a solution of a feasible schedule, and the second index (*computational cost*) calculates the accumulated steps. In this equation, "*step* (i)," "*start*," and "*iteration_in_convergence*" respectively indicate the steps counted in i iterations, the start iterations, and the iterations when the value of the total scheduling time converges through repetitions that attempt to find times shorter than the initial placement. This convergence is recognized when the total time shows the same value in some particular iterations. Furthermore, computational costs for repairing anomalies can be calculated by setting *start* to the iterations when anomalies make the schedule change.

3.2 Task planing for truss construction

3.2.1 Problem Description

In the task planning for truss construction with multiple space robots, we employ a robot which has only one arm in order to reduce its weight [†]. This means that each robot can only hold either a beam or a welding tool to combine/weld beams, which are the basic components of a truss. In a concrete truss construction with these robots, an example in the first several steps is shown in Fig.3. In this figure, the black circle with the solid line, the mesh circle, the double circle, and the dashed line respectively indicate a robot with its own beam, a robot without a beam, the space station, and the location for the truss that will be constructed. Note that all robots are supposed to have their own welding tools and thus robots without beams can weld beams by holding welding tools.

(1) Two robots hold their own beams and go to the beam constructing location.

(2) Two robots with their own beams arrive at the beam constructing location and set the desired angle between the beams. The robot without a beam goes to the welding location.

[†]In space, it is important to reduce the weight of robots because launching costs are quite expensive.

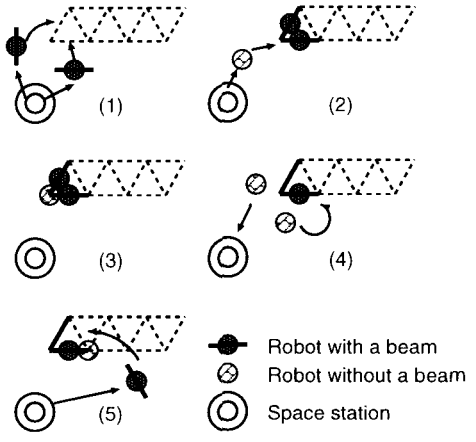


Figure 3: **Truss Construction**

(3) The robot without a beam arrives at the welding location and welds the beams.

(4) After welding, the robot with a beam on the left side returns to the station and the robot that welds the beams goes to another welding location.

(5) Another robot that has its own beam goes to the beam constructing location.

In addition, robots get into deadlocked situations when either all or none of the robots hold their own beams, or when some robots cannot go to the beam constructing or welding locations because other robots wait in the course of the target location.

3.2.2 Problem Setting

In this task, each robot is designed as an agent in OCS, and each learns to acquire an appropriate sequence of actions that minimizes the truss construction steps. In detail, robots have 11 primitive actions such as holding a beam, or moving toward a beam constructing location. Furthermore, robots can only recognize the situations of their neighbors.

As a concrete problem setting, all robots start at the space station and learn whether they hold their own beams or not. After beams are welded, robots that hold or weld beams learn again whether to go to other welding locations to weld the next beams or whether to return to the station to get other beams. When robots complete a truss construction or get into a deadlocked situation, all robots evaluate their own sequences of actions according to the current situation (completion or failure). Then the robots restart from the space station to acquire more appropriate sequences of actions that take fewer steps. In this cycle, one *step* is counted when all robots perform one primitive action, and one *iteration* is counted when robots complete to a truss construct or get into a deadlocked situation.

3.2.3 Index of Evaluation

In this task, the following two indexes are evaluated:

- Goodness = *truss construction step*
- Task completion rate

The first index (*goodness*) evaluates a solution that is the same viewpoint in the crew task scheduling, and the second index (*task completion rate*) evaluates how robots reconfigure cooperation among robots when there are anomalies. In particular, this rate is calculated as the average of the task completion numbers in a certain range of iterations. Note that the viewpoint of the second index is similar to that of the computational costs for repairing anomalies in the crew task scheduling.

4 Simulation

4.1 Experimental Design

A simulation investigates the rescheduling and reorganization abilities of OCS when anomalies occur. In the crew task scheduling, six types of anomalies shown in table 1 are introduced into a schedule of 10 jobs, and the results of rescheduling from the current schedule are compared with the results from the beginning. Since all constraints cannot be satisfied unless anomalies are removed, feasible schedules cannot be found. For example, a job that requires a link cannot be completed as long as a link is down. From this fact, this paper supposes a certain duration of anomalies. That is, both the start and end times are decided in each anomaly.

Table 1: **Type of anomalies**

Type	Anomaly	Content
1	Crew sick	A crew cannot perform experiments
2	Power down	The max size of power decreases
3	Link down	A link cannot be used
4	Machine A down	Machine A cannot be used
5	Machine B down	Machine B cannot be used
6	Type 1+2+3+4+5	Integration of 5 anomalies

In the task planning for truss construction, on the other hand, the four operations shown in table 2 are performed after the two robot groups A and B acquire some division of works while keeping their division of work, and the results after operations are compared with the results before operations. In particular, two groups construct their trusses from the same space station, thus they affect each other. Furthermore, each group is composed of five robots. Finally, the robot added in the "addition" operation is a new one which has not yet learned, and the failed robot in the "failure & removal" operation is an inoperative robot that behaves at random.

Table 2: **Operation**

Operation	Content
Addition	One robot is added to group A
Removal	One robot is removed from group A
Exchange	One robot in group A is exchanged with one robot in group B
Failure & Removal	One robot in group A fails and is removed

4.2 Experimental Results

Table 3 shows both the total scheduling time and the accumulated steps required in rescheduling for anomalies. The values are calculated both from the beginning and from the current schedule after anomalies occur. All results are averaged from five different examples of each anomaly type †. For instance, the duration time and anomaly start time of "link down" are different in each example.

Table 3: **Total scheduling time and accumulated steps**

Type	Total scheduling time		Accumulated steps	
	From the beginning	From current schedule	From the beginning	From current schedule
1	29.4	30.2	241.2	14.0
2	32.2	33.6	557.4	11.8
3	33.6	33.8	700.6	43.4
4	34.6	32.4	1581.8	16.4
5	32.2	29.4	1116.2	13.8
6	35.8	37.2	3204.2	38.0

Next, Table 4 and Fig. 4 respectively show the truss construction steps and the task completion rate, and compare the results before and after operations. Since this paper shows the change in the task completion rate, the values of Table 4 and Fig. 4 are obtained from one result. However, we have confirmed that the tendency of results does not change drastically with other examples or different random seeds. Furthermore, all operations except for "failure & removal" are performed in 117 steps, and "failure & removal" is performed in 117 and 417 steps. As shown in Table 4, the truss construction steps before operations in group A is smaller than those in group B because the location of truss A is nearer the station than that of truss B.

5 Discussion

(1) Rescheduling ability of OCS

The following discussions based on Table 3 suggest that OCS has a rescheduling ability that provides good feasible schedules quickly.

†This corresponds to the average of five situations with different random seeds in one example.

Table 4: **Truss construction steps**

Operation	Group A		Group B	
	Before	After	Before	After
Addition	232	208	313	313
Removal		310		310
Exchange		235		315
Failure & Removal		441		311

- **Total scheduling time** from a current schedule for each anomaly type is almost the same as the scheduling time from the beginning. This tendency does not change with the number of anomalies, even if constraints in a schedule become hard as the number of anomalies increases. Based on the fact that OCS finds good feasible schedules just from the current schedule, OCS has a mechanism for providing the appropriate rules for each job. Since these rules are acquired in just 103 accumulated steps † in the case of without anomalies, OCS is effective for practical and engineering use.
- **Accumulated steps** from the current schedule in each anomaly type is much smaller than those from the beginning (even if 103 accumulated steps which are needed for making a schedule in advance are added to the results from a current schedule). This effectiveness increases as the number of anomalies increases. Based on this fact, OCS provides a feasible schedule quickly in the case of rescheduling. Furthermore, this schedule is easy understandable for schedulers because most parts in the original schedule remain.

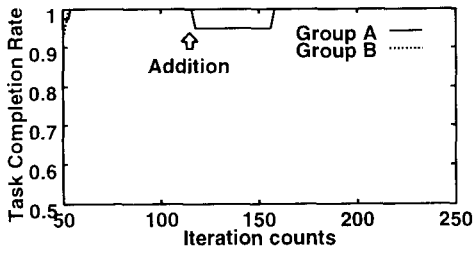
(2) Reorganization ability of OCS

The following discussions based on Table 4 and Fig. 4 suggest that OCS has a reorganization ability that keeps or recovers efficiency in tasks.

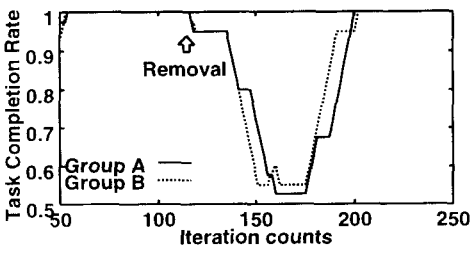
- **Addition** makes the truss construction steps in group A decrease from 232 to 208 steps because OCS enables an added robot to acquire the appropriate actions that are used to cooperates with the five original robots ‡. This keeps the same task completion rate.
- **Removal** makes the truss construction steps in group A increase from 232 to 310 steps because one robot is removed. This result can be understood by considering the fact that the truss construction steps increase as the number of robots decreases. Since an effective division of work

†103 accumulated steps can be calculated in about 3 seconds with a personal computer (Pentium 200MHz CPU).

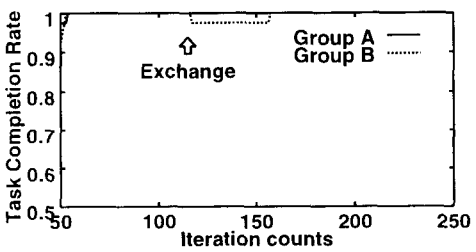
‡Basically, the truss construction steps decrease as the number of robots increases. However, this is based on the assumption that an added robot never fails to cooperate with others appropriately.



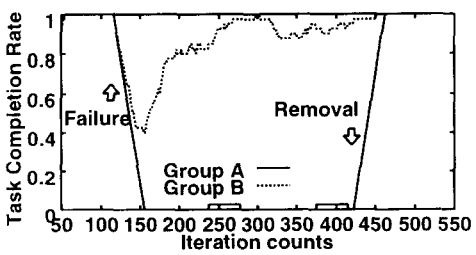
(a) **Addition**



(b) **Removal**



(c) **Exchange**



(d) **Failure & Removal**

Figure 4: **Four operations**

in group A is broken by removing a robot, the remaining robots must re-acquire new actions (which means to shift to a new division of work) to cooperate with each other again. This not only makes the task completion rate in group A decrease but also affects behaviors of group B. Thus, the task completion rate of group B decreases and increases according to the change in group A.

- **Exchange** does not drastically change the truss construction steps in group A, because OCS enables the exchanged robots to modify their actions to cooperate with other groups. This keeps the same task completion rate.
- **Failure & Removal** make the truss construction steps in group A increase from 232 to 441 steps due to the same reason as in "removal". However, the task completion rate of group B recovers more quickly than that in the "removal" case. This is because group A does not shift the original division of work to a new division of work due to the unfixed actions of the failed robot. Thus, the division of work in group B is not affected much by group A.

  However, one may think that OCS is not so useful in terms of fault tolerance because the rate of group A becomes almost 0 after one robot fails and does not recover until the failed robot is removed. However, this rate depends on the design of the actions of the robots. For example, robots do not get into the deadlocked situations mentioned in section 3.2.1 if we design actions that go back to the station to release a beam or other actions that move to another place while a few time. However, we *cannot guarantee* to design appropriate and indispensable actions in advance, especially *in space tasks*. Therefore, we must consider cases in which some robot groups confront unexpected situations, and we also must consider how other groups complete the tasks with being affected by those robots that confront unexpected situations. From this sense, OCS has the potential to recover the task completion rate when there are anomalies in some robot groups.
- **Truss construction steps in group B** does not change drastically because group B is not directly operated by the four operations.

(3) Possibility of OCS

In the task of space shuttles, schedulers appropriately assign jobs for each crew. However, there is a limitation to schedulers for space stations in which crews from different countries perform many experiments. This is because (1) there are more crew members on a space station than on a space shuttle and (2) experiments can be performed through 24 hours according to the time zone of each country. This situation obviously causes unexpected anomalies frequently. Even in such a case, OCS proposes good feasible schedules quickly. Furthermore, crews sometimes want to change constraints like job order because they know their jobs best, and these kinds of requirements occur asynchronously. In this case, OCS also provides this chance just by allowing crews to set their preferences for job constraints [†]. In particular, this property of OCS leads to effective coordination between crews and schedulers. At least, the hard work of schedulers is reduced to some extent.

In addition, cooperation among countries is indispensable in space stations. However, this is often

[†]Even if crews set their own preferences, OCS does not always satisfy these because the main aim of OCS is to improve total (organizational) performance according to the concept of organizational learning.

difficult because (1) jobs for each country are scheduled by each country's scheduler and (2) a sudden change in schedules affects other schedules, especially when the same instruments are used. Even in such a cases, OCS provides schedules that recover efficiency in jobs.

(4) Integration of three components

Although the effectiveness of OCS is shown through the above discussion, one may wonder if all three components ("learning mechanisms," "rule based systems with evolutionary approaches," and "multiagent approaches") are really needed to prove the effectiveness. In answer to this question, we have previously shown the effectiveness of integrating "learning mechanisms" and "rule based systems with evolutionary approaches" in OCS [Takadama 99a]. Furthermore, we have also shown the effectiveness of a "multiagent approach" integrated with the above two components. This was done by comparing the results of OCS with those of a model of the Michigan approach [Holland 78], which is one of conventional models in LCS and which employs the above two components in a centralized approach [Takadama 99b].

6 Conclusion

This paper has the explored possibilities of our organizational learning model and has shown its rescheduling and reorganization abilities through examples of the crew task scheduling in a space shuttle/station and the task planning for truss construction with multiple space robots. The main results are summarized as follows: (1) Our model provides good feasible schedules quickly in the case of rescheduling, and it keeps the computational cost for rescheduling low; (2) Plans generated by our model keep or recover efficiency in tasks when robots are added, removed, or exchanged among robot groups; and (3) The integration of (a) learning mechanisms, (b) rule based systems with evolutionary approaches, and (c) multiagent approaches is effective in rescheduling/replanning problems.

Future research will include an exploration of effective components, such as the above three properties, and will investigate their integrated effectiveness in scheduling and planning domains.

References

[Aarts 89] E. Aarts and J. Korst: *Simulated Annealing and Boltzmann Machines*, Jon Wiley & Sons, 1989.

[Argyris 78] C. Argyris and D.A. Schön: *Organizational Learning*, Addison-Wesley, 1978.

[Cohen 95] M.D. Cohen and L.S. Sproull: *Organizational Learning*, SAGE Publications, 1995.

[Fujita 96] S. Fujita and V.R. Lesser: "Centralized Task Distribution in the Presence of Uncertainty and Time Deadlines," *The Second International Conference on Multiagent Systems (ICMAS'96)*, pp. 95–102. 1996.

[Goldberg 89] D.E. Goldberg: *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989.

[Grefenstette 88] J.J. Grefenstette: "Credit Assignment in Rule Discovery Systems Based on Genetic Algorithms," *Machine Learning*, Vol. 3. pp. 225–245, 1988.

[Holland 78] J.H. Holland and J. Reitman: "Cognitive Systems Based on Adaptive Algorithms," in *Pattern Directed Inference Systems*, D.A. Waterman and F. Hayes-Roth (Eds.), Academic Press, 1978.

[Iima 99] H. Iima, T. Hara, N. Ichimi, and N. Sonnomiya: "Autonomous Decentralized Scheduling Algorithm for a Job-Shop Scheduling Problem with Complicated Constraints," *The 4th International Symposium on Autonomous Decentralized Systems (ISADS'99)*, pp. 366–369, 1999.

[March 91] J.G. March: "Exploration and Exploitation in Organizational Learning," *Organizational Science*, Vol. 2, No. 1, pp. 71-87, 1991.

[Osman 96] I.H. Osman and J.P. Kerry: *Meta-Heuristics: Theory and Applications*, Kluwer Academic Publishers, 1996.

[Russell 95] S.J. Russell and P. Norving: *Artificial Intelligence: A Modern Approach*, Prentice-Hall International, 1995.

[Takadama 98a] K. Takadama, S. Nakasuka, and T. Terano: "Printed Circuit Board Design via Organizational-Learning Agents," *Applied Intelligence*, Vol. 9, No. 1, pp. 25–37, 1998.

[Takadama 98b] K. Takadama, S. Nakasuka, and T. Terano: "Multiagent Reinforcement Learning with Organizational-Learning Oriented Classifier System," *The IEEE 1998 International Conference On Evolutionary Computation (ICEC'98)*, pp. 63–68, 1998.

[Takadama 99a] K. Takadama, T. Terano, K. Shimohara, K. Hori, and S. Nakasuka: "Making Organizational Learning Operational: Implication from Learning Classifier System, " *Computational and Mathematical Organization Theory (CMOT)*, 1999, to appear.

[Takadama 99b] K. Takadama, T. Terano, K. Shimohara, K. Hori, and S. Nakasuka. "Can Multiagents Learn in Organization? ~ Analyzing Organizational-Learning Oriented Classifier System ~," *The 16th International Joint Conference on Artificial Intelligence (IJCAI'99) workshop on Agents Learning about, from and with other Agents*, 1999, to appear.

[Tamaki 99] H. Tamaki, M. Ochi, and M. Araki: "Introduction of a State Feedback Structure for Adjusting Priority Rules in Production Scheduling," *Transaction of SICE (the Society of Instrument and Control Engineers)*, Vol. 35, No. 3, pp. 428–434, 1999, (in Japanese).

[Zhang 95] W. Zhang and T.G. Dietterich: "A Reinforcement Learning Approach to Job-shop Scheduling," *The 14th International Joint Conference on Artificial Intelligence (IJCAI'95)*, pp. 1114–1120, 1995.

[Zweben 94] M. Zweben, B. Daun, and M. Deale: "Scheduling and Rescheduling with Iterative Reaper," In *Intelligent Scheduling*, M. Zweben and M.S. Fox (Eds.), Morgan Kaufmann Publishers, Chapter 8, pp. 241–255, 1994.