TR-H-273

Scheduling with Organizational Learning Agent.

Masakazu WATABE (UnivWollongong/ATR-HIP), Keiki TAKADAMA and Katsunori SHIMOHARA

1999.6.30

ATR人間情報通信研究所

〒619-0288　京都府相楽郡精華町光台2-2　TEL: 0774-95-1011

ATR Human Information Processing Research Laboratories

2-2, Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0288, Japan

Telephone: +81-774-95-1011

Fax : +81-774-95-1008

Scheduling with Organizational Learning Agents

○ **Masakazu Watabe**
The University of Wollongong/
ATR HIP Research Labs.
North Fields Avenue, Wollongong
NSW 2522 Australia
mw08@uow.edu.au

Keiki Takadama
ATR HIP Research Labs.
2-2 Hikaridai, Seika-cho
Soraku-gun, Kyoto
619-0288 Japan
keiki@hip.atr.co.jp

Katsunori Shimohara
ATR HIP Research Labs.
2-2 Hikaridai, Seika-cho
Soraku-gun, Kyoto
619-0288 Japan
katsu@hip.atr.co.jp

Abstract

The new possible application of our organizational learning model is revealed in this study. The model consists of several concepts including Reinforcement Learning (RL), Rule-based system (including Generation/Exchange of rules), Multiagent system. This model, called Organizational-Learning Oriented Classifier System, is applied to the scheduling of space crews' tasks. As well as the normal scheduling, we conduct the scheduling with anomalies aimed at measuring the effectiveness of this model in unexpected situations. Furthermore, we discuss the effective way to design the agents' actions in the domain. The series of experiments shows the acceptable performances of the model; it provides practically feasible schedules at low computational cost and with completion times of all tasks in both expected and unexpected situations.
Keywords: organizational-learning oriented classifier system, learning classifier system, task scheduling/rescheduling, multiple learning agents, rule design

1 Introduction

Scheduling is a mostly common area where Artificial Intelligent (AI) is applied. There are several approaches taken in this domain such as Operational Research (OR), AI and Simulation. OR approaches are classified into Heuristic and Meta-Heuristic including Genetic Algorithm (GA), Neural Network and Simulation Annealing (SA). AI approaches are classified into Expert System and Domain-specific heuristics. Ways to apply those approaches will be varied, depending on the characteristics of applied scheduling domains. Several approaches, described above, are practically applied to and successfully operational in the mobile vehicle industry [Parunak 99], and some are prototyped at research laboratories [Fukui 97, Hara 98]. The approaches in the scheduling domain has several advantages as well as dis-advantages. For instance, as a well-known fact, the GA approaches have a number of types where the system generates a large number of primitive schedules as genotype. Incidentally, this leads to inflexibility in the case of re-scheduling needed because those facts eventually result in high computational cost. Thus they are not realistic methods under strict resource/technical constraints.

Adaptive behavior is an area lately focused by some researchers in robotics as well as other areas [Stone 97] and is defined as: "Learning produces changes within an agent that over time enable it to perform more effectively within its environment" [Arkin 98]. Some mechanisms of adaptation such as learning mechanism, rule-based systems with evolutionary approach and multiagent approach successfully show its effectiveness against the previous approaches in the scheduling domain [Ito 98, Kim 98]. For instance, Zhang applied a reinforcement learning

to the NASA space shuttle payload processing tasks where Zweben's method based on SA originally has been in use [Zweben 94] and shows a better performance [Zhang 95]. In addition, Tamaki's production rule-based system with the evolutionary mechanism implies the rule-based system's adaptability into dynamic environment [Tamaki 99]. Furthermore, Fuji and Iima's distributed multiagent approaches show the multiagent system's ability to schedule and reschedule [Fujita 96, Iima 99]. Those examples clearly indicates that the possibilities of those adaptation mechanisms are useful against the problems with the conventional approaches.

Although those mechanisms can supplement one another, those approaches are developed separately and never considered its effectiveness as an integrated system. Therefore, we propose our model, Organizational Learning Model [Takadama 98a, Takadama 98b, Takadama 98c, Takadama 99a and Takadama 99b]. This model consists of the adaptation mechanisms that employ a reinforcement learning, rule-based system with evolutionary approach and multiagent approach. And we applied this model to the space shuttle/station task scheduling domain. In this domain, several approaches have been taken and experimentally put in practical use [Muscettola 97]. However, there is no definite way to produce feasible schedules at acceptable computational cost. This model produces the feasible schedules at relatively low cost in terms of time to compute. In addition, this model has been applied to and successfully shows its effectiveness in several other domains such as the truss construction [Takadama 98c], Computer Aided Design (CAD) on the circuit board design [Takadama 98a] and Pentomino (a playing domain) [Takadama 99b].

This paper is organized roughly as follows: Section 2 briefly introduces Organizational Learning in the conceptual as well as computational models, Section 3 describes the overview of the system, Section 4 describes the scheduling domain as well as the system design, Section 5 experiment on the scheduling and effect of the rule design; Section 6 discusses the implication of the experiment results, Section 7 suggests considerations in scheduling crews' tasks in practical, and finally the conclusion in Section 8.

2 Organizational Learning

2.1 Four Loop Learning in Organizational Learning

Organizational Learning (OL) is a study area in organization and management science. Its purpose is to analyze the characteristics of organization. OL proposes four types of learning loop [Kim 93, Argyris 78]. Those types are at individual/organizational level and can be classified as single/double.

- **Individual Single-Loop Learning** improves performance within an individual norm.
- **Individual Double-Loop Learning** improves performance by the change of an individual norm.
- **Organizational Single-Loop Learning** improves performance within an organizational norm.
- **Organizational Double-Loop Learning** improves performance by the change of an organizational norm.

2.2 Four Loop Learning in Computation

Those conceptual norms are converted into the computational models. Note the interpretation of these norms will vary from person to person. Due to this fact, no details of implementation in code is described here. But the assumptions are made to those norms.

- **Individual norm** is implemented by individual knowledge
- **Organizational norm** is implemented by organizational knowledge

And the following assumptions are for our OCS

- **Individual knowledge** is implemented by a set of rules
- **Organizational knowledge** is implemented by a set of individual knowledge

The organizational learning in OCS is defined as *"Learning that includes four types of the computational loop learning".*

3 Learning Classifier System and Organizational-learning oriented Classifier System

Learning Classifier System (LCS) is originally developed by Goldberg and Holland [Goldberg 89, Holland 78], and mainly consists of three mechanisms: (1) Rule-based system; (2) Reinforcement Learning; (3) and Genetic Algorithm (GA). Rule-based system provides a problem solving function. Reinforcement Learning provides improved performances. GA provides rule generation/exchange functions. Those functions contribute to adaptation of the system into its environment. Our Organizational-learning oriented Classifier System (OCS) consists of a number of Learning Classifier Systems (LCSs) and combines Organizational Learning (OL) [Argyris 78, March 91, Cohen 95] to improve individual performances as well as the performance of organization of those LCSs.

3.1 System Architecture

The system consists of a number of agents. The agents have the following major characteristics:

1. All the agents have the same architecture inside (see Fig.1).
2. They do not have communication amongst them. The rule exchange is not considered as communication here.
3. They recognize other agents as part of its environment.
4. They perceive the state of local environment in stead of that of global one.
5. They are not controlled by a central control system/agent. They are autonomous.

< Problem Solver >

- **Detector and Effector** change a part of its environmental state into an agent-understandable internal state (IF part of production rule) and into output as an action (THEN part of the rule) respectively [Russell 95].

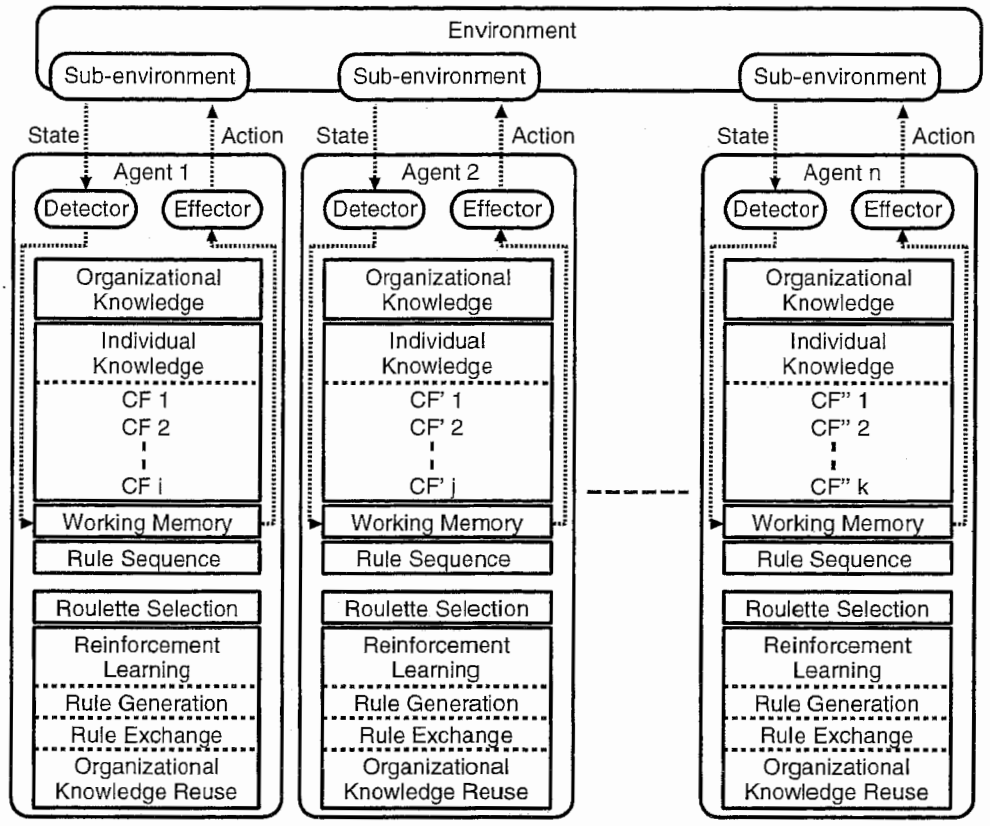

Figure 1: **OCS Architecture**

< **Memory** >

- **Individual knowledge memory** stores a set of if-then rules, called Classifiers (CFs). Each rule in the storage has the strength as well.
- **Working memory** stores fired rules.
- **Rule sequence memory** stores a sequence of fired rules that will be reinforced.

< **Other Mechanisms to Note** >

- **Roulette selection** selects one rule from the rules that match with its environment state. The process in which a rule is chosen is based on the roulette selection.
- **Reinforcement learning, rule generation, rule exchange, and organizational knowledge reuse mechanisms** are converted into the computational model from the conceptual model of OL.

3.2 Rule Design

An agent has a number of production rules. A rule in the agent consists of three parts that are IF, THEN and STRENGTH. The IF part has several parts in it. The number of parts are corresponding to the number of environmental states (*i.e.*, Constraints). The THEN part has an action. This part is fired according with its environmental state. The STRENGTH part is the weight to express the efficiency of the rule and initially is given the same values amongst

the rules. The value of strength is changeable, responding to the performance of the agent. The strength of consecutive rules in individual agent, which lead to a deadlock situation and worse system performance, are given a negative reward. And the strength of those rules in all the agents contribute to better system performances, on the other hand, are given a positive reward. Its environmental state, surrounding agents, is digitized into 0 or 1 on the flags of the IF part through the detector. And if a rule matched, the rule fires its predefined action through the effector.

3.3 Adaptation Mechanisms in OCS

- **Reinforcement learning mechanism:** The RL mechanism helps agents to select those effective rules in its problem solving. In OCS, the larger the value of strength is, the more chances the rule is chosen. Those rules with higher strength are considered as more effective in solving problem. A profit sharing method [Grefenstette 88] is employed and modified for the RL in our OCS [Takadama 98b].

- **Rule Generation mechanism:** This mechanism generates new rules when no rule in the storage matches the environment state (*i.e.*, Condition). When the number of rules reaches its MAX-CF(maximum number of CFs), the rule with the lowest strength is overwritten by the new rule. The strength of the new rule is given a Zero value.

- **Rule Exchange mechanism:** Agents exchange their rules with others at predefined time(step) intervals. The exchange is occurred between two agents. The pair is randomly selected. For instance, in Figure 2, Agent X and Y are selected. Their CFs are sorted by its strength (if a CF is located at upper, the strength of the CF is higher). And Agent X and Y exchange rules. The number of rules to be exchanged is proportional to the number of the existing rules. Those agents , for this example, decide three rules to exchange. The Agent X's first three CFs at upper are exchanged for the Agent Y's first three CFs at lower and vice versa. The assumption of the need of exchange is that some rules might not be effective for some agents but those rules might be effective for other agents in its problem solving.

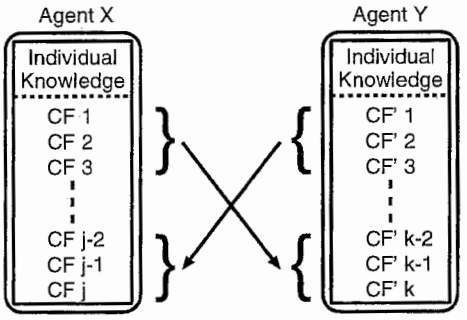

Figure 2: **Rule Exchange Mechanism**

4 Scheduling in Space Shuttle/Station Crews' Tasks

4.1 Problem Description

This section describes the task scheduling for space shuttle/station crews. The number of crews is six including two Mission Specialists (MSs) who are in charge of experiments and four Payload

Specialists (PSs) who support experiments. A job has a series of sub-tasks and can be divided into up to five sub-tasks. Those sub-tasks must be performed in accord with its order. Each sub-task requires one to six personnel. In addition, the duration of sub-tasks (up to 6 units of time) and the number of crews required for a sub-task is determined in advance. Some sub-tasks are only assignable to either PSs or MSs, and some are to the pair of a MS and a PS.

1. **Power of Space Shuttle/Station:** Each job requires some amount of electric power that ranges from 0 to 100 percentage. At a unit of time, the power usage is limited to less than 100 percentage.

2. **Link to the ground station:** Some jobs require the link with the ground station. The link is operational once in a unit of time. And at every 10 units of time, the link can not be operational.

3. **Machine A:** Some jobs require Machine A. Machine A is operational once in a unit of time. Machine A includes a computer, voice recorder, and etc..

4. **Machine B:** Some jobs require Machine B. The rest is the same as above.

5. **Order amongst jobs:** Each sub-tasks must be done one after another according with its order.

6. **Assignment of a sub-task:** Five types of assignment to a sub-task are set: (1) anybody can be assignable; (2) PS(s) is assignable; (3) a particular PS and any of the rest are assignable; (4) a particular MS and any of the rest are assignable; and (5) the pair of a MS and a PS is assignable. One of those types is assigned to a sub-task.

7. **Sub-tasks** can not be located in one unit of time, that is, there is no over-lap between sub-tasks.

Note that one unit of time is 10 minutes. 72 units (one unit = 10 min.) of time equals to 24 hours.

4.2 Rule Design in Scheduling

The design of the rules, that is, the way the agents percept its environment and take appropriate actions will vary from domain to domain. In the process of trial and error, the best rule design will be created. For our system, as a general design of the rules, the following design is employed.

- **IF Part:** The IF part is divided into eight sub-parts: (1) Overlap; (2) Power; (3) Link; (4) Machine A; (5) Machine B; (6) Order; (7) Assignment Type; (8) and Deadlock Avoidance. The 8th item is used to avoid a deadlock situation in which an agent, violating some constraint over time, can not move to any other positions. Those constraints become flags in the IF part. If an agent violates any constraint(s), the flag(s) become ON(1), otherwise, OFF(0).

- **THEN Part:** The THEN part has an action in each rule. Totally, 15 actions are designed. Those actions can be classified mainly into the following two types. In addition, different characteristics of Neighbors Search are created.

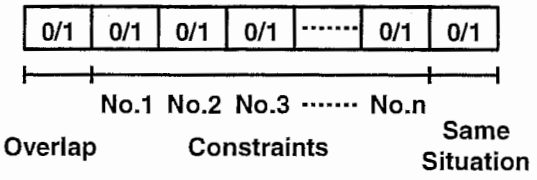

Figure 3: **IF Part**

- **Neighbors Search:** When agents violate constraints, to avoid this situation over time, agents move to the right and left in turn to find a place that satisfies all constraints. We divide this type of the actions into two characteristics, called Altruistic and Selfish. The Altruistic actions try to find a place where the agent does not have influence on other agents' constraint(s). On the other hand, the Selfish actions try to avoid only the agent's own constraint(s) so that it might have impact on other agents' constraint(s). Neighbors Search has 14 actions.
- **Left Search:** Since Neighbors Search is not designed to minimize the length of schedule, this search is designed. If all constraints are satisfied, an agent is able to move to the left-most position. That results in minimizing the length of the schedule. One Left Search action is created.

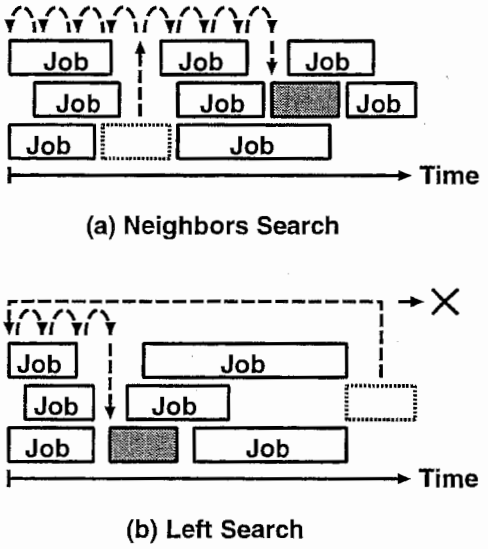

Figure 4: **THEN Part**

4.3 Initial Environment/Setting and Structure of Program

At each unit of time, One link, Machine A, Machine B, and 100 percentage of electric power are usable. But the link is not operational at every 10 units of time due to the orbit of the space craft. In the situation with anomalies, some of those items are not operational. There are a number of jobs in a day. Each job is assigned to an agent that has a LCS in it. All the agents are placed at random on the schedule table (72 units of time $\times$ The number of crews) in the first place. The agents, then, evaluate their local environment, fire a rule that matches its environment state and move to the positions according with the search actions in turn.

The cycle in which all the agents perform their actions is called "STEP". The agent with lower sub-task number starts his action first before the others. At some intervals, the rules

are exchanged between agents. And the strength of the rules that violate any constraints are temporary weakened at the end of one agent's step. Those rules are recovered after satisfying the constraints. On the other hand, when scheduling starts to converge(the system produces the constant length of the schedule with a number of steps), the "ITERATION" is counted. And all the rules are given rewards at the end of each iteration. Three types of rewards are used:

```
"Zero" if
 Min Sch < Pre Sch <= Min Sch + Mid4
"Plus" if
 Pre Sch < Min Sch or Min Sch + Mid4 <= Pre Sch < MidPoint
"MINUS" if
 MidPoint <= Pre Sch

 Mid4 = (Max Sch - Min Sch) / 4, MidPoint = (Max Sch + Min Sch) / 2
 Min: Minimum, Max: Maximum, Pre: Present, Sch: Schedule time, Mid: Middle
```

And finally if the system consecutively produces the schedule with convergence for more than 10 times, scheduling is terminated. The rough pseudo-code is described as follows:

```
Pseudo-code:
 Initializing
  Iteration-Loop{
   Put agents randomly on schedule table
   Step-Loop (Each agent){
    Evaluate local environment
    Fetch rule from database
    if no rule
       Create new one
    Take action
    if constraint(s) violated
       Restrict agent's action to rules
    if CROSS-OVER time comes
       Do CROSS-OVER
    if converged
       Go out the Step-Loop
   } /*Step-Loop*/
   Give rewards to agents according to performances
   if converged more than 10 times
      Go to End
  } /*Iteration-Loop*/
  End
```

5 Simulation

5.1 Scheduling

5.1.1 Experiment

The experiment is conducted to measure the effectiveness of OCS in the normal scheduling. The number of Classifiers is initially 25 and is able to increase up to 50. Four cases of the schedules

are prepared and the best (human-made) schedule times are shown in Table 1. The number of sub-tasks in each schedule is as follows:

- Case 1: 10 sub-tasks
- Case 2: 10 sub-tasks
- Case 3: 11 sub-tasks
- Case 4: 12 sub-tasks

The variables, such as the amount of Power and Link usage, in each case differ. And the different time scales are used in evaluating two types of the schedules in Table 1. Seconds are for computer and Minutes are for human. A personal computer with Pentium 200MHz CPU is employed.

5.1.2 Evaluation

One measurement is used for the performance in the normal scheduling.

- Gap = (Computer-made schedule) - (Human-made schedule)
- Scheduling time

The length of the human-made schedules are subtracted from that of computer-made schedules. This equation is employed to demonstrate that OCS produces the feasible schedules. The second measurement is Scheduling time that indicates how long it takes to produce the schedules. The human-made schedules are created by the author.

5.1.3 Result

The result of the experiments is shown in Table 1. The gaps between the computer-made and human-made schedules are accumulated and divided by the number of the cases. It does not show the large difference between two types. Those are almost the same schedule time. The difference of those two types of the schedule is 2 units of time on average. Noticeably, the case 3 has no difference. On the other hand, the scheduling time shows the large difference. Clearly, the human takes longer than the computer does. And the scheduling time of the human is increased as the schedule becomes more complex (*i.e.*, the number of jobs is large). However, the computer does not show that fact.

Table 1: **Computer-made and Human-made Schedule**

CASE	Computer-made Schedule		Human-made Schedule		GAP
	Schedule time	Time (Sec.)	Schedule time	Time (Min.)	
1	35	8	34	45	+1
2	28	15	25	47	+3
3	36	10	36	50	+0
4	23	7	19	65	+4
Average	—	—	—	—	+2

5.2 Re-scheduling

5.2.1 Experiment in Anomalies

The experiments are conducted to measure the effectiveness of OCS in the situations in which some anomalies happen. 5 anomalies are defined: (1) a crew is sick; (2) the electric power is down by some amount; (3) the link is not operated; (4) Machine A is not usable; (5) and Machine B is not usable; (6) the combination of those five anomalies. Those anomalies last one to 10 units of time. And five sets of those six types of the anomalies are created. In each set, the duration of the anomalies and start/end times are randomly generated. The number of Classifiers is initially 25 and is able to increase up to 50. In the simulation, the schedule with 10 sub-tasks is selected, and two cases of the experiments are conducted and are compared. The first is that an anomaly happens from the beginning. The other experiment is that an anomaly happens after the convergence in the normal situation.

5.2.2 Evaluation

Two indexes are used in the experiments.

- Schedule time
- Accumulated steps = $\sum_{i=start}^{iteration\_in\_convergence} step\ (i)$

The first index is calculated by adding all the schedule time in convergence and the sum is divided by the number of anomaly types. The second one is calculated by adding the number of steps until the convergence and the sum is divided by the number of anomaly types. "$step(i)$", "$start$" and "$iteration\_in\_convergence$" indicates the steps counted in i th iteration, the start of the iteration, and the first iteration of the convergence respectively.

5.2.3 Result

Table 2 shows the result of two cases of the experiments. The left-hand side of the column is for the first experiment and the right-hand side is for the second one in Schedule Time and Accumulated steps respectively. Note that all the figures are accumulated and divided by the number of the anomaly file sets, and since the anomalies happen after the convergence (in the normal situation), the figures on the Accumulated Step column of the current schedule could be added another 103 steps that are taken to converge (it takes about 3 seconds with Pentium 200MHz CPU). Even if 103 steps added to the column of After the convergence, the accumulated steps is still less than From the beginning.

5.3 Design of rules

5.3.1 Rule design and System performance

The question arises from the design of the rules. That is, "Is it true that there is a way to solve the problem more effectively? And how the effective and ineffective actions have impact on the performance of the system?". This experiment is designed to examine the system performances with the different combination of actions. As introduced, the three different actions, (L)eft and (A)ltruistic and (S)elfish, are used. We set seven different cases as shown below. In the cases from 1 to 3, each individual action is taken by the agents no matter what situations they are in. And from the case 4 to 6, the combination of two rules is taken. For the case of 4 and 5, unless

Table 2: **Schedule time and Accumulated steps**

Type	Schedule time		Accumulated steps	
	From the beginning	After the convergence	From the beginning	After the convergence
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

all the constraints are satisfied (Left-most search is taken), the other action is executed. For the case 6, the agents takes Altruistic actions unless the dead lock occurred. At last, for the case 7 the combination of three types of the actions are examined. Five schedule samples in that the values of parameters differ are used to take the average for the comparison in each case.

- Case 1, 2, 3: L, A, S
- Case 4, 5, 6: LA, LS, AS
- Case 7: LAS

5.3.2 Evaluation

Two indexes are used in the experiments, and these are the same ones in the previous experiments.

- Schedule time
- Accumulated steps= $\sum_{i=start}^{iteration\_in\_convergence} step\ (i)$

5.3.3 Result

Fig.5 shows the result of the system performance in each case. The white bars indicate the schedule time and the blank bars indicate the computational cost for each case. The cross sign at (L)eft-most search indicates that the system could not reach the convergence. For the other cases, the results of the samples are averaged.

6 Discussion of Results

6.1 Scheduling with OCS

Table 1 clearly shows the ability of OCS in scheduling. Importantly, there is not considerable difference between the human- made and computer-made schedules, which implies that OCS is able to create the feasible schedules. In addition, the time to create the schedules clearly differs between two types. The computer performs faster than the human. And noticeably the duration of scheduling is increased as the schedules become complicated for the human. On the other hand, this fact is not seen for the computer. Since this schedule system is designed in

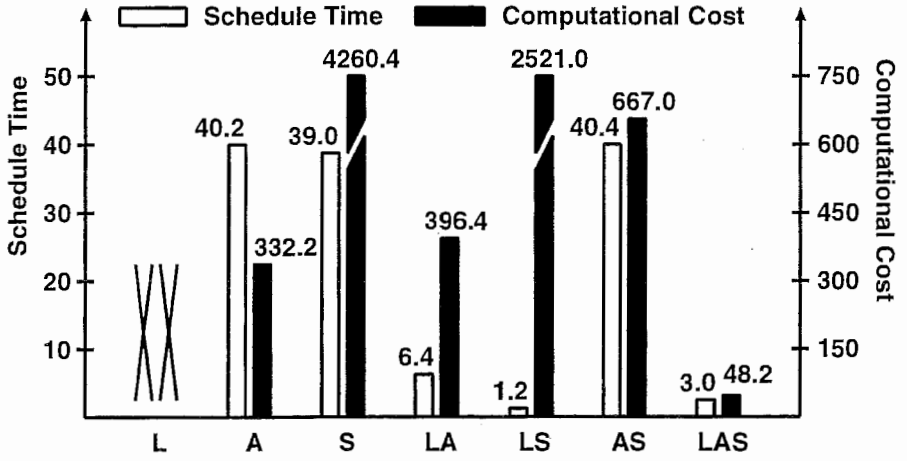

Figure 5: **Schedule Time and Computational Cost**

practical space use where the there are more jobs to be performed and the limited time to do some activities under restricted resources, those comparisons between two types indicate that OCS is more effective than human in scheduling.

6.2 Re-scheduling with OCS

Table 2 clearly shows the ability of OCS in re-scheduling at reasonable computational cost. The major findings are (1) there are not the large differences with the schedule time between the first and second experiments in Schedule Time and (2) the number of the accumulated steps in the second experiment is fewer than in the first one of Accumulated Steps. The first fact indicates the re-scheduling ability of OCS. As well as from the beginning, OCS is able to create the new schedules after the convergence. The second fact indicates that the rules that have been already reinforced contribute to produce the schedule with fewer steps than those that are not reinforced. And as a matter of fact, the fewer the number of steps is, the lower the computational cost is.

6.3 Design of rules

Several points can be discussed from Fig.5. First, (L)eft-most search itself can not produce the schedule while the two other actions reach the convergence with infeasible schedule times. Second, the combination of two rules at least is able to produce the schedule. Furthermore, the (S)elfish actions take more time than the (A)ltruistic. The reason of this lengthy computational time of the (S)elfish actions attributes to the exploration factor. That is, this type of actions acts as explorer in its huge search space of the outcomes. So, (S)elfish actions have the potential to find better/worse schedule time. On the other hand, the (A)ltruistic actions take the steady computational time due to the exploitation factor. That is, this type of actions facilitates the paths that have been already explored. Thus, it reaches the same, final output steadily. However, the potential of improvement on the outcome is smaller than the other factor described. Third, the combination of three actions shows the best outcome. This indicates that each type of actions is able to supplement the shortages amongst them so that it has the highest potential to reach the best outcome on the computational cost as well as the schedule time. The result on Fig.5 clearly shows those points.

7 Considerations for More Practical Use in Space Shuttle/Station Scheduling

For more practical use of this scheduling, the following topics must be considered.

1. There are usually more than six crews in a station.
2. Experiments are performed according to the time of each country on the board.
3. Human schedulers at each country create their own schedules to follow.
4. There are power games amongst countries in executing missions.
5. Re-scheduling might be necessary at any point of time in a day.
6. There are usually more constraints on the board.
7. The number of jobs might be changeable (the number of jobs is reduced/increased).
8. The number of anomalies might be more than described in the previous section.

8 Conclusion

Our organizational learning model shows the effectiveness in scheduling as well as in re-scheduling. The model consists of three adaptation functions that employ the learning mechanism, rule-based system with the evolutionary function and finally multiagent system. Three types of the experiments are conducted to examine the scheduling and re-scheduling abilities of the system, and the impact of the rule design. The series of intensive experiments shows major achievements. The feasible schedules are created in the short time and the computational time of the computer is constant although the cases of the experiments become complicated. In addition, the results of the second experiments show that the rescheduling with anomalies is possible at low computational cost and OCS effectively reinforces its rules used to achieve the objects. And finally, the third experiment shows that the rule design surely has an impact on the system performance. Future researches involve cancelation/addition of jobs, the limitation of local perception by an agent and finally re-design this scheduling for more practical use as they are mentioned in the previous section.

Acknowledgment

Most of all, I would like to thank to Katsunori Shimohara who is the head of the department 6 and who gave me this opportunity. I hope I was not his big headache. And I would like to thank to Keiki Takadama who gave me all instructions to follow and helped me with programming with a lot of patient. Readers should be noted that this paper is done thanks to his support. Otherwise, I could not go through this report. And finally I would like to thank to those who support me for the last 6 months at ATR.

References

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

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

[Arkin 98] R. C. Arkin: *"Behavior-Based Robotics: Intelligent Robots and Autonomous Agents,"* The MIT Press, 1998.

[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.

[Fukui 97] A. Fukui, H. Iima and N. Sonnomiya: *"Autonomous Decentralized Scheduling System for an Operation Assignment Problem,"* Journal of System, Control and Information, Vol. 10, No. 3, pp.107-115, 1997, (in Japanese).

[Goldberg 89] D. E. Goldberg: *"Genetic Algorithm 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, 1998.

[Hara 98] T. Hara, N. Ichimi, H. Iima and N. Sonnomiya: *"Evaluation of an Autonomous decentralized Scheduling Algorithm Using a Simulator,"* The 8th Intelligent System Symposium, 1998, (in Japanese).

[Holland 78] J. H. Holland and J. Reitman: *"Cognitive Systems Based on Adaptive Algorithms,"* 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 Jop-Shop Scheduling Problem with Complicated Constraints,"* The 4th International Symposium on Autonomous Decentralized Systems (ISADS'99), pp.366-369, 1999.

[Ito 98] T. Ito and T. Shintani: *"Persuasion Based on Exchanging for Cooperative Scheduling,"* The Transactions of IEICE (the Institute of Electronics, Information and Communication Engineers), Vol.J18-D-I, No.9, pp.1099-1106, 1998. (in Japanese)

[Kim 93] D. Kim: *"The Link between Individual and Organizational Learning,"* Sloan Management Review, Fall, pp.37-50, 1993.

[Kim 98] G. H. Kim and C. S. G Lee: *"Genetic Reinforcement Learning Approach to the Heterogeneous Machine Scheduling Problem,"* IEEE Transactions on Robotics and Automation, Vol.14, No.6, pp.879-893, 1998.

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

[Muscettola 97] N. Muscettola, P. P. Nayak, B. Pell and B. C Williams: *"Remote Agent" To Boldly Go Where No AI System Has Gone Before,"* http://ic-www.arc.nasa.gov/ic/projects/Executive/team/barney/ Autonomy/ autonomy.html, 1997.

[Parunak 99] H. V. D. Parunak: *"Industrial and Practical Application of DAI,"* In the book of Multiagent System, G. Weiss (Eds.), Massachusetts Institute of Technology, Chapter 9, pp.377-421, 1999.

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

[Stone 97] P. Stone and M. Veloso: *"Multiagent Systems: A Survey from a Machine Learning Perspective,"* http://www.cs.cmu.edu/ pstone/, 1997.

[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 98c] K. Takadama, K. Hajiri, T. Nomura, K. Shimohara and S. Nakasuka: *"Organizational Learning model for Adaptive Collective Behaviors in Multiple Robots,"* Advanced Robotics, Vol.12, No.3, pp.243-269, 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.