Can Multiple Learning Agents Implement Responsiveness, Conflict Resolution, and Collaboration Between Autonomous Scheduling Systems and Human Operators?

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

Shinichi Nakasuka
Univ. of Tokyo
7–3–1 Bunkyo–ku
Tokyo 113–8656 Japan
nakasuka@space.t.u–tokyo.ac.jp

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

Abstract

This paper focuses on multiple learning agents in crew task scheduling problems and explores their capabilities of responsiveness, conflict resolution, and collaboration in interactions between autonomous scheduling systems and human operators. A careful investigation of these capabilities has revealed the following implications: (1) multiple learning agents provide responsiveness that enables them to quickly modify their schedules in cases of unexpected anomalies; (2) when human operators change the conditions of jobs in the schedule, conflicts are smoothly resolved through interactions among agents; (3) human operators can collaborate with agents by introducing preliminary or rough schedules into autonomous scheduling systems.

Keywords: multiple learning agents, crew tasks scheduling, responsiveness, conflict resolution, collaboration

Introduction

Recently, the importance of autonomy is recognized among mission technologists and scientists to address unpredictable and complex situations in space. Based on this understanding, several autonomous systems such as Hubble Telescope (Muscettola 93) have been successfully implemented in actual missions. This kind of system contributes to closed-loop control situations where experiments or other work can be completed without human operations. However, many systems still require human support especially those in scheduling domains. For example, an appropriate decision regarding a schedule change is needed when the system meets unexpected situations. In such a case, the relationship between autonomous scheduling systems and human operators is important for overcoming trouble.

However, how can we implement appropriate autonomous systems for the above case? We do not have a good answer to this question. To address this issue, we start by focusing on multiple learning agents in autonomous scheduling systems and explore their possibility through an evaluation of the following three capabilities: (1) responsiveness, (2) conflict resolution, and (3) collaboration, all of which are important aspects in an interaction between autonomous scheduling systems and human operators. In particular, the first capability is required in the case of anomalies to quickly provide new acceptable schedules and minimize the time loss. The second capability is needed to smoothly modify schedules according to mission changes or other schedule change requirements. Finally, the third capability is required to undertake the partially completed work of human operators and help them to reduce their burdens.

This paper is organized as follows. The next section describes a multiagent model, and then shows the crew task schedule on a space shuttle or station. Simulations and discussions are made in the following sections. Finally, our conclusions are given in the last section.

*Paper submitted to the 2nd NASA International Workshop on Planning and Scheduling for Space

Organizational-Learning Oriented Classifier System

An Organizational-learning oriented Classifier System (OCS) (Takadama 99a) is one of the multiagent models that introduces the concept of organizational learning (OL) (Argyris 78; Cohen 95) studied in organization and management science.*[1] Since OCS is based on Learning Classifier Systems (LCSs) (Goldberg 89), OCS has similar mechanisms to those of LCSs, but the big difference*[2] between OCS and LCSs is that (1) OCS has a multiagent learning architecture, and (2) OCS addresses the division of work in multiple agents, both of which are difficult in conventional approaches.

Agents

OCS is composed of many LCSs as shown in Fig. 1. In OCS, agents (jobs in this paper) are implemented by their own LCSs which are extended to introduce four kinds of learning mechanisms reinterpreted from OL.*[1] In order to solve problems that cannot be solved at an individual level, agents divide given problems by acquiring their own appropriate *functions* through interaction among agents. According to this approach, the *aim* of the agents is defined as finding appropriate functions. Since these functions are acquired through the change in each agent's rule sets and through the change in the strength *[3] of rules, a *function* is defined as a rule set in OCS.

*[1]A detailed introduction or reinterpretation of the concepts of OL is discussed in (Takadama 99a).

*[2]A detailed difference is described in (Takadama 99b).

*[3]Strength is defined as the worth or weight of rules.

Architecture

As shown in Fig. 1, each agent in OCS has the same architecture that includes the following components.

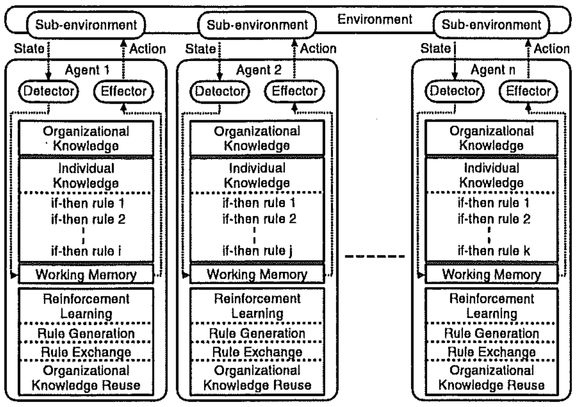


Figure 1: **OCS architecture**

< **Problem Solver** >

- **Detector** and **Effector** translate a part of an environment state into the internal state of an agent and derive actions based on the internal state (Russell 95), respectively.

< **Memory** >

- **Organizational knowledge memory** stores a set comprising each agent's rule set as organizational knowledge. In OCS, this knowledge is shared by all agents. Furthermore, organizational knowledge represents knowledge on the division of work because each agent's rule set derives the role of each agent.
- **Individual knowledge memory** stores a rule set (a set of *if-then* rules including a *strength* factor) as individual knowledge. In OCS, agents independently store different rules.
- **Working memory** stores the recognition results of sub-environmental states and also stores the internal state of an action of fired rules.

< **Mechanisms** >

- **Reinforcement learning, rule generation, rule exchange, and organizational knowledge reuse mechanisms** are reinterpreted from the four kinds of learning in OL [*4].

Learning in OCS

(1) Reinforcement learning mechanism: In OCS, the reinforcement learning (RL) mechanism enables agents to acquire their own appropriate actions required to solve given problems. In particular, RL supports the learning of the appropriate order of fired rules by changing the strength of the rules. Specifically, OCS employs a *profit sharing* method (Grefenstette 88) that reinforces a sequence of all rules when agents complete given tasks.

[*4] Details are described in (Takadama 99a).

(2) Rule generation mechanism: The rule generation mechanism in OCS creates new rules when none of the stored rules matches the current environmental state. In particular, when the number of rules is `MAX_RULE` (the maximum number of rules), the rule with the lowest strength is removed and a new rule is generated. In the 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.

(3) Rule exchange mechanism: In OCS, agents exchange rules with other agents at a particular time interval (`RULE_EXCHANGE_STEP`[*5]) in order to acquire effective rules that cannot be acquired by agents themselves. In this mechanism, a particular number ((the number of rules)×`GENERATION_GAP`[*6]) of rules with low strength values are replaced by rules with high strength values between two arbitrary agents. However, rules that have a higher strength value than a particular value (`BORDER_ST`) are not replaced to avoid unnecessary operations that increase communication costs. The strength value of replaced rules are reset to their initial values.

(4) Organizational knowledge reuse mechanism: Finally, agents in OCS store a set comprising each agent's rule set (individual knowledge) as knowledge on the division of work when they most effectively solve given problems,[*7] and agents reuse this knowledge when they solve other problems. For example, when n agents most effectively solve problems, a set comprising each agent's rule set is stored as $\{$RS (1), RS (2), $\cdots$, RS (n)$\}$, where RS(x) is the rule set for the x-th agent. In OCS, this set is called organizational knowledge[*8] and is updated through problem solving.

Supplemental setup

In addition to the above mechanisms, a particular number (`FIRST_RULE`) of rules in each agent is generated at random in advance, and the strength values of all rules are set to the same initial value.

Crew Task Scheduling

Problem Description

The crew task scheduling of a space shuttle or station is a job-shop scheduling problem where many jobs for

[*5] This step is defined later.

[*6] The ratio of operated rules.

[*7] Although it is difficult to generally define efficiency, agents, for example, recognize that they solve a given problem most effectively by measuring a "good solution" or a "small computational cost."

[*8] Note that agents cannot use both individual and organizational knowledge at the same time because the latter knowledge can be utilized by all agents as learned rules.

the crews must be scheduled under hard resource constraints. The goal of this problem is to find feasible schedules that minimize the total schedule execution (TSE) time of all jobs. We selected this domain because (1) this problem can be considered as a multiagent problem when one job is assumed to be one agent, and (2) a systemization of this problem is required to support schedulers at ground stations. In this task, there are several missions composed of jobs, and these jobs should be assigned while satisfying the following constraints to accomplish the missions.

1. **Power of space shuttle or station:** Each job requires a particular power (from 0% to 100%) in the experiments, but the summation of the power of all jobs at any time must not exceed 100%.
2. **Link to the ground station:** Some jobs require a link to the ground station, but only one job at a time can use it. Due to the orbit of the spacecraft, none of the jobs can use the link during certain times.
3. **Machine A:** Some jobs need to use machine A in the experiments, but only one at a time job can use it. Some examples of machines are computers, voice recorders, and so on.
4. **Machine B:** This is the same constraint as that for machine A.
5. **Execution order of jobs:** In a mission unit, jobs have an execution order, but some jobs in a mission may be only partially ordered, which means that some jobs may have the same order.[*9] In comparison with jobs, there is no order/priority among missions, and thus jobs in a certain mission are scheduled with jobs in other missions considering their respective execution 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 unit of a job, one of the following crew assignment types is set: (a) Anybody, (b) PS only (but the concrete crew is not specified), (c) One specified PS with somebody, (d) One specified MS with somebody, and (e) Combination of PS and MS (but the concrete crews are not specified). These crew assignments are based on actual space shuttle missions.

In addition to the above six constraints, "the length" and "the required number of crew members" for each job are also decided beforehand.

Problem Setting

In the task, each job is designed as an agent in OCS, and each job learns to acquire an appropriate sequence of actions that minimizes the TSE time. Specifically, jobs can only observe local situations in the range of the length of their jobs[*10] except for the TSE time calculated from a location of an agent that is set at the latest time in the schedule.[*11] Based on this local observation, the rules of each job are designed to only consider local information including jobs' own primitive actions. Thus, only related (neighbor) agents can recognize the change made by another agent.

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

Evaluation Criteria

The following two indexes are evaluated in the task, and the *performance* is defined as a criterion which considers the two indexes.

- Solution = $TSE\ time$
- Computational cost = $\sum_{i=1}^{n} step\ (i)$

The first index (*solution*) evaluates the execution time of a feasible schedule, and the second index (*computational cost*) calculates the accumulated steps. Especially in the latter equation, "*step* (i)" and "n", respectively, indicate the steps counted in i iterations and the number of iterations when the converged TSE time shows the same value in particular iterations.

Simulation

Experimental Design

A simulation investigates the abilities of multiple learning agents in the following three cases. All cases are tested with crew task schedules that involve 10 jobs in five missions.

- **Case 1: Responsiveness**
  This case addresses an anomaly situation where one crew member cannot perform experiments for a certain duration due to illness, and investigates the result after this anomaly using a comparison between

[*9]The order described here is most simple one. Other representations are required to handle partially order in general.

[*10]Examples of the local informations include information of an overlap, a power, a link, and so on.

[*11]The TSE time as global information is updated synchronously every after all agents perform one primitive action such as movements that satisfy power constraints.

the performance from the first placement of agents (jobs) and that from the current placement. When starting from the first placement, all agents are placed at their initial locations and then move their locations by considering the parts of the anomaly. When starting from the current placement, on the other hand, all agents move their locations from the current schedule that satisfies all constraints except for the parts of the anomaly. Specifically, only agents that do not satisfy constraints start by changing their locations in the schedule, and then other agents move their locations when their constraints are violated. In anomaly cases like that described above, new acceptable schedules must be obtained as quickly as possible. Therefore, both agents from the first and current placements utilize learned rules that are acquired before such situations in order to minimize the time loss.

- **Case 2: Conflict resolution**
  This case addresses a conflict situation caused by the change in the job's execution order from lower to higher, and investigates the result after such conflict using a comparison between the performance from the first placement of agents and that from the current placement. Since solutions may become worse due to rules that do not consider the conflict situations, agents from the first placement do not utilize learned rules while those from the current placement utilize learned rules acquired before the conflict situations.
- **Case 3: Collaboration**
  This case addresses a collaboration situation where human operators (the authors in this paper) introduce preliminary or rough schedules into agents, and investigates the result after such collaboration using a comparison between the performance of not using learned rules and that of using rules. Since a start placement is decided according to preliminary or rough schedules, both performance are calculated from the current placement of agents. Furthermore, agents have no learned rules in this case, and thus agents learn rules without collaboration and utilize them in the case of using learned rules.

In the above three cases, the learning mechanisms in agents process every iteration, which means the time when the TSE time converges. These mechanisms are turned off when the converged TSE time shows the same value in the particular iterations. Note that the TSE time is guaranteed to converge because in the end all agents are unable to find locations where the TSE time becomes small without overlapping as agents are placed at early times in the schedule.

Furthermore, in the case of not using the learned rules, agents start with randomly generated rules set and apply the learning procedure. In the case of using the learned rules, on the other hand, agents utilize the learned rules and apply the learning procedure. Note that the learned rules in the above three cases are not utilized on the same problem in the true sense but are utilized on a similar problem. The reasons for this are summarized as follows: (1) in the anomaly or conflict situations, the structure of the problem changes when an anomaly or a conflict occurs because the learned rules generated before anomaly or conflict situations do not consider such situations; (2) in the collaboration situation, on the other hand, the type of problem without collaboration is different from that with collaboration because the learned rules without considering collaboration do not always cover the situations of the collaboration.

Finally, since the learned rules are implemented by a set comprising each agent's rule set which includes only local information, these rules are assumed to be the organizational knowledge described in the section on OCS. In particular, considering the fact that organizational knowledge represents knowledge on the division of work, the learned rules can be applied to similar or other problems like those in this simulation.

Experimental Setup

Variables in OCS are set as follows: `FIRST_RULE` (the number of initial rules) is 25, `MAX_RULE` (the maximum number of rules) is 50, `RULE_EXCHANGE_STEP` (the interval steps for rule exchange operations) is 10, `GENERATION_GAP` (the percentage of operated rules) is 10%, and `BORDER_ST` (the lowest strength of the rule not for removal) is -50.0 *[12].

Experimental Results

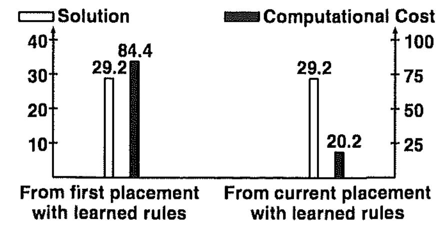
Fig. 2 shows the solution (the TSE time) and the computational cost (the accumulated steps) in the crew task scheduling problem. In detail, Figs. 2 (a), (b) and (c) show the results of responsiveness, conflict resolution, and collaboration, respectively. In this figure, the left axis and the white box indicate the average of the "solution", and the right axis and the black box indicate the average of the "computational cost". Furthermore, all results are averaged from five different random seeds. The results suggest that (1) none of the solutions in the three figures change drastically even whether the agents start from the first or current placement or whether the agents utilize the learned rules or not; (2) all computational costs, on the contrary, become small when the agents start from the current location or utilize the learned rules in comparison with the computational costs of others.

Discussion

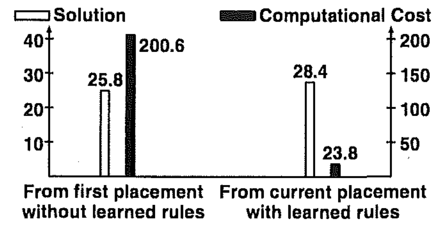
(1) Why Multiple Agents?

Figs. 2 (a) and (b) show that the computational costs become small while the solution is kept at the same level when the agents start from the current placement after anomaly or conflict situations. This is because

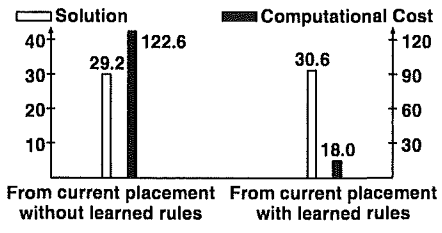
*[12]Note that the tendency in results does not drastically change according to the parameter setting.



(a) **Responsiveness**



(b) **Conflict Resolution**



(c) **Collaboration**

Figure 2: **Solutions and computational costs**

multiagent systems have the potential to be robust in dynamical environment changes caused by anomalies or conflicts. Specifically, multiple agents tend to explore a solution space according to independent decisions among agents, and this exploration contributes to a weakening or absorbance of an influence from anomalies or conflicts. This means that the multiagent environment frequently changes due to changes made by other agents, and thus agents simply address anomaly or conflict situations by considering them as one kind of environmental change. However, solutions never converge just by exploring a solution space. Therefore, agents in OCS decide their actions according to the local information as a main activity, but manage to converge by sharing only one global information (*i.e.*, the TSE time).*[13] Based on this implementation, agents explore a solution space until the TSE time converges. Thus, the solution level is maintained even if the agents start from the current placement, and this start contributes to a reduction in the computational costs by preventing from large modifications like those from the first placement.

Furthermore, agents in OCS have no way of knowing what the other agents are doing due to a multiagent approach. This seems that agents may not satisfy their constraints. However, agents find their appropriate places that satisfy their constraints, by checking whether their constraints are satisfied or not through local information.

*[13]Other scheduling metrics that involve global calculations can be also handled by OCS like in the TSE time.

(2) Why Learning Agents?

Figs. 2 (b) and (c) show that the computational costs become small while the solution is kept at the same level when the agents utilize the learned rules. This is because the learned rules acquired before conflicts or acquired without considering collaborations work as factors of both *reactiveness* and *deliberation*. In detail, the if-then part in the rules links input conditions and output actions, and thus it works as a kind of reactive planning (Agre 87) that works as a factor of reactiveness. The strength part in the rules, on the other hand, creates a chain among rules by changing the worth of rules,*[14] and thus it works as a kind of classical planning (McDermott 78) that works as a factor of deliberation. From these factors, it is quite important for the agents to acquire both indispensable if-then combinations and an appropriate rule chain in order to reduce the computational costs while keeping the solution at the same level.

(3) Effectiveness of Multiple Learning Agents

From the above discussions, we arrive at the conclusion that both *multiple* and *learning* are important aspects for agents to effectively address (a) responsiveness, (b) conflict resolution, and (c) collaboration, all of which are done through an interaction between autonomous scheduling systems and human operators. Although this result does not cover all practical cases, our previous research have found that multiple learning agents in OCS is also effective in other domains and large scale problems (Takadama 99a; Takadama 99b). Here, this section specifies the advantage of multiple learning agents in more detail by investigating the relationships among the (a), (b), and (c) cases as shown in Fig. 3.

- **Applicable range of multiple learning agents:** Cases (a), (b) and (c) compare different areas, respectively. For example, the responsiveness case compares the result from the first placement with that from the current placement, while the collaboration case compares the result of not using the learned rules with that of using the rules. These different focuses indicate the wide applicable range of multiple learning agents. In particular, unexpected situations often break predetermined schedules in space domains, and thus a wide applicable range has the potential to cope with such situations.

*[14]A rule with a high strength value is included in a rule chain, while a rule with a low strength value is excluded in the chain.

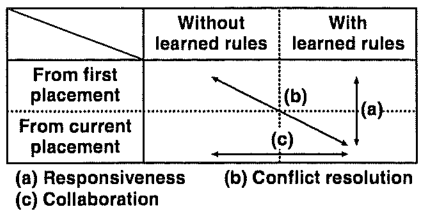


Figure 3: **Relationship among three cases**

- **Another axes of reactiveness and deliberation:** As described in the previous section, both reactiveness and deliberation are key factors to address in the cases (a), (b) and (c). However, this axis of reactiveness and deliberation is considered only from the viewpoint of the learning aspect. Here, we add another two axes of reactiveness and deliberation as follows: (1) a decision whether the agents start their locations from the current or the first placement, and (2) a decision whether the agents utilize the learned rules or not. These are because the agents from the current placement or the agents utilizing the learned rules contribute to finding feasible schedules quickly, while the agents from the first placement or the agents without utilizing the learned rules contribute to finding better schedules by exploring a wide range of search space. From this discussion, the three axes of reactiveness and deliberation are embedded in multiple learning agents. Considering the fact that these axes are tradeoff relationships and that they affect by each other, it is generally difficult to implement appropriate systems due to a lot of unpredictable factors. However, the results in this simulation suggests that agents staring from the current placement with the learned rules are robust in such tradeoffs.

(4) Possibility of Multiple Learning Agents

The above advantage of multiple learning agent approaches contributes not only to scheduling domains but also to multiple satellites or multiple space robots to address more complex and difficult tasks. In these cases, each satellite or robot is assumed as one agent. Furthermore, cooperation among many schedules can be implemented by assuming one schedule as one agent. This kind of cooperation is quite important in an international space station where a lot of schedules are performed asynchronously.

As another possibility, multiple learning agents have the potential to provide effective heuristic knowledge by investigating the trace of agent's movement when good schedules are found with small computational costs. An example of heuristic knowledge shows that jobs which require a long time must be placed first. Note that this kind of heuristic knowledge cannot be found from a global viewpoint but from a local one. This means that we must investigate what kind of rules each agent learn. As a result, the acquisition of heuristic knowledge provides the operators with explanations of the system's behavior.

Conclusion

This paper focused on multiple learning agents in crew task scheduling problems and explored their capabilities of responsiveness, conflict resolution, and collaboration in interactions between autonomous scheduling systems and human operators. The main results are summarized as follows: (1) multiple learning agents provide responsiveness that enables them to quickly modify their schedules in cases of unexpected anomalies; (2) when human operators change the conditions of jobs in the schedule, conflicts are smoothly resolved through interactions among agents; (3) human operators can collaborate with agents by introducing preliminary or rough schedules into autonomous scheduling systems.

Future research will include a further investigation of "understandability" for human operators. A comparison with benchmarks or conventional methods that involve the scheduling theory (Brucker 95) must be made. Furthermore, an evaluation in realistic scenarios is also needed in future work.

References

P.E. Agre and D. Chapman (1987). "Pengi: A implementation of a Theory of Activity," *The 6th National Conference on Artificial Intelligence (AAAI'87)*, pp. 268–272.

C. Argyris and D.A. Schön (1978). *Organizational Learning*, Addison-Wesley.

P. Brucker (1995). *Scheduling Algorithm*, Springer-Verlag.

M.D. Cohen and L.S. Sproull (1995). *Organizational Learning*, SAGE Publications.

D.E. Goldberg (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley.

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

D. McDermott (1978). "Planning and Action," *Cognitive Science*, Vol. 2, pp. 71–110.

N. Muscettola and S.F. Smith (1993). "Constraint-Directed Integration of Scheduling and Planning for Space-Based Observatory Management", *The 7th Annual Space Operations, Applications and Research Symposium (SOAR-93)*.

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

K. Takadama et al. (1999a). "Making Organizational Learning Operational: Implications from Learning Classifier System," *Computational and Mathematical Organization Theory (CMOT)*, Kluwer Academic Publishers, Vol. 5, No. 3, pp. 229–252.

K. Takadama et al. (1999b). "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.*