Can Multiagents Learn in Organization? ~ Analyzing Organizational-Learning Oriented Classifier System ~

○ **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

Takao Terano
Univ. of Tsukuba
3-29-1, Otsuka, Bunkyo-ku
Tokyo 112-0012 Japan
terano@gssm.otsuka.tsukuba.ac.jp
Tel: +81-3-3942-6855
Fax: +81-3-3942-6829

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

Koichi Hori
Univ. of Tokyo
4-6-1 Komaba, Meguro-ku
Tokyo 153-8904 Japan
hori@ai.rcast.u-tokyo.ac.jp
Tel: +81-3-3481-4484
Fax: +81-3-3481-4585

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

An Organizational-learning oriented Classifier System (OCS) is an extension of the Learning Classifier Systems (LCSs) to multiagent environments. Unlike the conventional multiagent systems in the literature, which use specific and elaborate techniques, OCS integrates four mechanisms from multi-strategic standpoints that are derived from the concept of organizational learning (OL) in organization and management science. Intensive experiments on a complex scalable domain have revealed that (1) the integration of the four learning mechanisms in OL is effective in solution and computational cost; (2) OCS finds good solutions at less computational cost than conventional LCSs; and (3) the concept of OL has the potential to overcome limitations of conventional multiagent methods or techniques.

Keywords: multiple learning agents, organizational learning, learning classifier system, multiagent category

1 Introduction

Recently, many research [Weiss and Sen, 1996; Weiss, 1997] have addressed problems faced in multiagent environments. Examples include reinforcement learning [Sutton and Bart, 1998], distributed problem solving in distributed artificial intelligence (DAI) [Gasser and Bond, 1988] and evolutionary computation [Goldberg, 1989] However, such research seems to have concentrated on improvements in particular methods or techniques in this domain.

Unlike these conventional studies, our Organizational-learning oriented Classifier System (OCS) [Takadama *et al.*, 1998a] extends Learning Classifier Systems (LCSs) [Goldberg, 1989] by integrating four mechanisms from multi-strategic standpoints. These mechanisms are derived from the concept of organizational learning (OL) [Argyris and Schön, 1978; March, 1991; Cohen and Sproull, 1995] in organization and management science. In our previous work, the effectiveness of OCS was shown by comparing the results of human experts in the printed circuit boards (PCBs) design problems of Computer Aided Design (CAD). However, we did not analyze why or how OCS can work well or how the concept of OL affects research on multiagents. To address these issues, this paper theoretically investigates OCS from the viewpoint of OL, experimentally compares the performance of OCS with those of the conventional LCSs, and empirically analyzes the characteristics of research on multiagents according to OL for clarifying which methods or components are needed for improving the collective performance in multiagent environments.

This paper is organized as follows. Section 2 starts by describing OL, and Section 3 explains OCS. An example for analyzing OCS is given in Section 4. Sections 5 and 6 respectively give simulations and a comparison to conventional LCSs. Related works are analyzed from the viewpoint of OL in Section 7. Finally, our conclusions are given in Section 8.

2 Organizational Learning

2.1 Four loop learning in organizational learning

Organizational learning (OL) has been studied in the context of organization and management science and is

*Paper submitted to IJCAI'99 Workshop on Agents Learning about, from and with other Agents

roughly characterized as organizational activities that improve organizational performance which cannot be achieved at an individual level. In particular, OL consists of the following four kinds of learning [Kim, 1993; Argyris and Schön, 1978]:

- **Individual single-loop learning** improves performance within an individual norm.
- **Individual double-loop learning** improves performance through the change of an individual norm itself.
- **Organizational single-loop learning** improves performance within an organizational norm.
- **Organizational double-loop learning** improves performance through the change of an organizational norm itself.

2.2 Reinterpretation of four loop learning

The categorization in the previous section stipulates that (1) there are individual and organizational levels in learning, and (2) each learning can be classified as a single or a double type. However, *norms* † in the above learning are not fully computational, which means that they can be implemented in many ways from a computational viewpoint. Based on this fact, this paper does not discuss how to implement these norms in detail, although this issue is quite important in terms of the collective performance in multiagent environments. Therefore, this paper starts by assuming the following norms.

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

Furthermore, the following two types of knowledge are assumed to support the above assumptions.

- **Individual knowledge** is implemented by a rule set.
- **Organizational knowledge** is implemented by a set of individual knowledge (rule set) units.

The above assumptions may seem to define only parts of organizational learning, but it is important to find new results which can be derived from the above assumptions. Based on this claim, this paper defines *computational organizational learning* as "learning that includes four kinds of reinterpreted loop learning."

3 Organizational-learning oriented Classifier System

An Organizational-learning oriented Classifier System (OCS) is GBML (Genetics-Based Machine Learning) architecture and OCS is composed of many Learning Classifier Systems (LCSs), which are extended to introduce the four kinds of reinterpreted loop learning mechanisms described in previous section.

3.1 Aim of agent and function

In OCS, agents are implemented by their own LCSs. 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. From 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, and thus a *function* is defined as a rule set (individual knowledge). 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.

3.2 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 not the state of the total environment.

< 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 and Norving, 1995], 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 and is called knowledge on the division of work.
- **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 with 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

†For instance, routines and weltanschauung are examples of an individual norm and an organizational norm, respectively.

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 loop learning in OL (Details are described later).

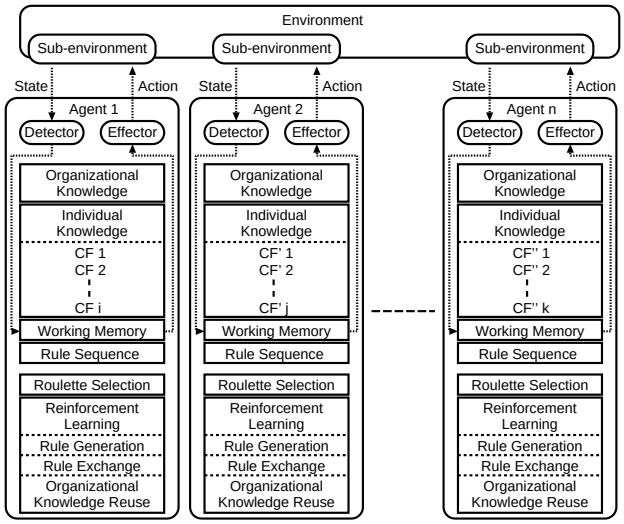


Figure 1: **Architecture of OCS**

3.3 Learning in OCS

Reinforcement learning mechanism

In OCS, the reinforcement learning (RL) mechanism enables agents to acquire their own appropriate actions which are required to solve given problems. In particular, RL supports agents to learn an appropriate order of the fired rules by changing the strength of the rules. Since this mechanism improves the problem solving efficiency at an individual level not by creating/deleting rules but by utilizing them while changing the order of the fired rules, it works as one kind of "individual single-loop learning," which is reinterpreted to improve performance within individual rules in computational organizational learning. In detail, this mechanism works as shown in Fig. 2, and is implemented by *profit sharing* [Grefenstette, 1988], which reinforces a sequence of rules at once when agents obtain some rewards †.

Rule generation mechanism

The rule generation mechanism in OCS creates new rules when none of the stored rules match the current environmental state as shown in Fig. 2. 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. Since this mechanism improves the problem solving range at an individual level by creating/deleting rules, it works as one kind of "individual double-loop learning," which is reinterpreted to improve the performance through the change of individual rules themselves in computational organizational learning.

As 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 same initial value. Furthermore, when 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.

Rule exchange mechanism

In OCS, agents exchange rules with other agents at a particular time interval (`GA_STEP` †) in order to solve given problems that cannot be solved at an individual level as shown in Fig. 2. Since this mechanism improves the problem solving efficiency at the organizational level not by creating/deleting a set comprising each agent's rule set but by utilizing it among the agents, this mechanism works as one kind of "organizational single-loop learning," which is reinterpreted to improve the performance within a set comprising each agent's individual rules in computational organizational learning.

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. 3, the CFs in each agent are sorted by order of their strength (upper CFs have high strength values). In this case, $CF_{j-2} \sim CF_j$ and $CF'_{k-2} \sim CF'_k$ are replaced by $CF'_1 \sim CF'_3$ and $CF_1 \sim CF_3$, respectively. However, rules that have higher strength 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.

Organizational knowledge reuse mechanism

Finally, agents in OCS store a set comprising each agent's rule set (individual knowledge) as their knowledge on the division of work when they solve given problems most effectively§, and reuse it when the `REUSE_FLAG` is `TRUE` ¶. For example, n number of agents most effectively solve problems by using their rule sets, a set com-

†The detail credit assignment in OCS was proposed in [Takadama *et al.*, 1998b].

†This step is defined in section 4.2.

‡The ratio of operated rules.

§Since efficiency depends upon the problems, it is difficult to generally define efficiency. Some examples indicate a "good solution" or "small computational cost." In addition, efficiency does not mean an optimal solution but means the best one since the agents have found through their experiments up to the present.

¶Although `REUSE_FLAG` can be set as `TRUE` in anytime, a reusing time in this stage of OCS is fixed at the iteration 0. This iteration is defined in section 4.2.

```
procedure reinforcement learning
  begin
    if problem is solved then
      for all agents do
        fired rules are reinforced;
  end

procedure rule generation
  begin
    for all agents do
      if no matched rules then
        begin
          if number of rules = MAX_CF then
            a rule with the lowest strength is deleted;
          a new rule is created;
          strength of a new rule is set to an initial value;
        end
  end

procedure rule exchange
  begin
    if mod (step, GA_STEP)=0 then
      for all pair of agents do
        for (number of rules)×GENERATION_GAP rules do
          if lowest strength of rule ≤ BORDER_ST then
            begin
              a rule with low strength is replaced by a rule
                with high strength between two agents;
              strength of a replaced rule is reset to an
                initial value;
            end
  end

procedure organizational knowledge reuse
  begin
    if REUSE_FLAG=TRUE then
      stored organizational knowledge is utilized;
    else if solution is best then
      begin
        if organizational knowledge is stored then
          stored organizational knowledge is deleted;
        current organizational knowledge is stored;
      end
  end
```

Figure 2: **Algorithms of the four learning mechanisms**

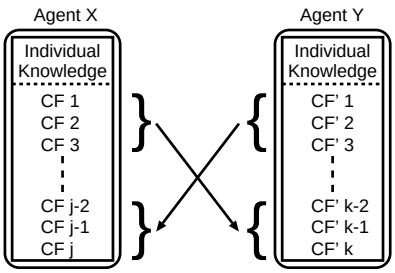


Figure 3: **Rule exchange**

prising each agent's rule set is stored as shown in Fig. 4. In particular, the rule sets already stored are replaced by new ones. In OCS, this set is called organizational knowledge and is represented by {RS (1), RS (2), $\cdots$, RS (n)}, where RS(x) is the rule set for the x-th agent and n is the number of total agents in the organization.

Since this mechanism improves the problem solving range at an organizational level by creating/deleting organizational knowledge, it works as one kind of "organizational double-loop learning," which is reinterpreted to improve performance through the change of a set comprising each agent's individual rules itself in computational organizational learning. Note that agents cannot use organizational knowledge directly because this knowledge is an organizational level knowledge (Organizational knowledge can be used after knowledge is divided into separate individual knowledge). This indicates that agents cannot use both individual and organizational knowledge at the same time. Furthermore, organizational knowledge is different from ordinary effective knowledge in a single LCS. This is because the former knowledge represents the division of work and is used in an unit of multiagent organization, while the latter knowledge is useful for all agents.

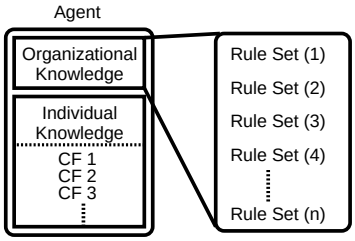


Figure 4: **Organizational knowledge reuse**

Relationships among the four learning mechanisms

A total algorithm of OCS follows the procedure as shown in Fig. 5. Briefly, organizational knowledge is reused if it is stored before agents solve a problem, and both the mechanisms of rule generation and rule exchange are executed until the problem is solved. After problem solving, both mechanisms of reinforcement learning and organizational knowledge reuse are executed, and agents continue to solve the same problem from the same initial situation until the solution is converged.

3.4 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 are generated at random, and the strength values of all rules are set to the same initial value.

3.5 Comparison between conventional LCSs and OCS

In the context of LCS, conventional LCSs can be roughly divided into two types: the Michigan [Holland and Reitman, 1978] and Pittsburgh [Smith, 1983] approaches.

```
procedure OCS
  begin
    iteration=0;
    REUSE_FLAG=TRUE;
    organizational knowledge reuse;
    REUSE_FLAG=FALSE;
    while solution is converged do
      begin
      step=0;
        while problem is solved do
          begin
            rule generation;
            rule exchange;
            step=step+1;
          end
        iteration=iteration+1;
        reinforcement learning;
        organizational knowledge reuse;
      end
  end
```

Figure 5: **Algorithms of OCS**

As shown in Table 1, the former approach includes many rules in a population, and these rules are evaluated by credit assignment. The latter approach, on the other hand, includes many rule sets in a population, and these rule sets are evaluated through elite selection. Compared to these two conventional LCSs, OCS includes many rule sets in a population, but the rules in the rule sets are evaluated by credit assignment. Although other differences between conventional LCSs and OCS can be found in both genetic operations (crossover, mutation, inversion) and a concept of OL as shown in Table 1, the big decisive difference is that OCS addresses the division of work in multiagents which is difficult for conventional approaches to do.

Table 1: **Comparison of characteristics between conventional LCSs and OCS**

	Michigan	Pittsburgh	OCS
Population	rule	rule set	rule set
Evaluation	rule	rule set	rule
	credit assignment	elite selection	credit assignment
Crossover	between rules	between rule sets	between rule sets
Mutation	rule	rule	—
Inversion	—	rule set	—
Concept of OL	a part †	a part †	whole
Problem solving	by LCS	by LCS	by multiple LCSs which divide works

† Detail analysis is discussed in section 7.

4 Pentomino Tiling Problem

4.1 Problem description

A pentomino is a figure that combines five squares as shown in Fig. 6 (a), and its tiling problem is to appropriately place the pentominos while minimizing the area that encloses all pentominos without any overlap. We select this domain because (1) this problem can be considered as a multiagent problem when one pentomino is assumed as one agent, (2) it is easy to increase/decrease the number of pentominos, (3) the minimum solution is known as shown in Fig. 6 (b), and (4) this problem can be directly applied to engineering domains such as printed circuit boards (PCBs) design problems in Computer Aided Design (CAD) that look for a part layout that minimizes the total wiring length (in this case, each part corresponds to each pentomino [Takadama *et al.*, 1998a]).

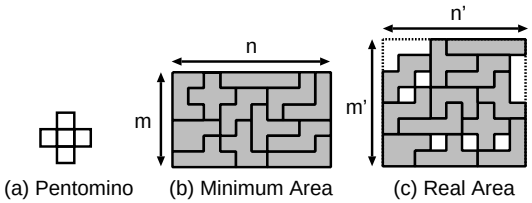


Figure 6: **Pentominos**

4.2 Pentomino design and problem setting

In this task, each pentomino is designed as an agent in OCS, and each pentomino learns to acquire an appropriate sequence of actions that minimizes the area enclosing all pentominos without any overlap. In detail, the pentominos have 17 primitive actions including stay, move, and rotation.

As a concrete problem setting, all pentominos are initially placed at random without considering any overlap, and therefore most pentominos overlap with each other. After this initial placement, the pentominos start to perform some primitive actions to reduce the overlap and to minimize the area that encloses all pentominos. When the size of this area converges without any overlap, all pentominos evaluate their own sequences of actions according to the size of the area. Then, the pentominos restart from the initial placement to acquire more appropriate sequences of actions for finding a smaller area. In this cycle, one *step* is counted when all pentominos perform one primitive action, and one *iteration* is counted when the size of the area converges without overlap.

4.3 Index of evaluation

In this task, the following two indexes are evaluated:

$$Goodness = \frac{Real\ area}{Minimum\ area} \quad (1)$$

$$Computational\ cost = \sum_{i=1}^{iteration\ in\ convergence} s(i) \quad (2)$$

The first index (*goodness*) evaluates a solution and shows how the current area, as shown in Fig 6 (c), is small compared with the minimum area in Fig. 6

(b). In this case, *goodness* is calculated in every iteration by $\frac{m' \cdot n'}{m \cdot n}$, where m', n', m and n are the lengths of the sides in the area. The next index (*computational cost*) calculates the accumulated steps. In this equation, "$s\ (i)$," and "*iteration_in_convergence*" respectively indicate the steps counted in i iterations and the iterations when the size of the area converges through repetitions in an attempt to find a smaller area from the initial placement. This convergence is recognized when the size of the area shows the same value in some particular iterations.

Table 2: **Real computational cost**

	Michigan	Pittsburgh	OCS
Every step	n_a agents† perform one action	pn_a agents† perform one action	n_a agents† perform one action
Every `GA_STEP`‡	crossover + mutation	—	crossover
Every iteration	credit assignment	crossover + mutation + inversion	credit assignment + organizational knowledge

† n_a indicates the number of agents in an organization. p indicates the number of LCSs that solve the same problem at the same iteration.

‡ `GA_STEP` in Pittsburgh approach can be considered as one iteration.

Basically, computational cost must include all content shown in Table 2. For example, when $cost1(i)$, $cost2(i)$, and $cost3(i)$ in OCS are respectively assumed as the cost of every step, the cost of every `GA_STEP`, and the cost of every iteration, three costs in an i iteration are calculated as follows.

$$\begin{aligned} cost1(i) &= n_a s(i) \cdot (n_r lc + k_a c) \\ cost2(i) &= n_a \frac{s(i)}{\texttt{GA_STEP}} \cdot (n_r glc) \\ cost3(i) &= n_a (s(i) \cdot c + n_r lc) \end{aligned}$$

In the above equations, n_a, $s(i)$, n_r, l, c, k_a, and g respectively indicate the number of agents, the number of steps in an i iteration, the number of rules, the length of a rule, the cost for one value (or bit) match, the coefficient that adjusts the cost for an action ($k_a >> 1$: In general, the cost of performing an action is much larger than c), and the generation gap ($0 < g < 1$). In particular, the first and second terms in $cost1$ respectively indicate the cost of rule matching and the cost of performing an action. In $cost3$, the first and second terms respectively indicate the cost of a credit assignment and the cost of storing organizational knowledge. These equations derive a relationship $cost1(i) >> \{cost2(i), cost3(i)\}$ that is also satisfied in both the Michigan and Pittsburgh approaches. Thus, the computational cost in an i iteration can be calculated as follows.

$$\begin{aligned} &Computational\ cost(i) \\ &\quad = Cost1(i) + Cost2(i) + Cost3(i) \\ &\quad \approx Cost1(i) \\ &\quad = C \cdot s(i) \end{aligned}$$

In this equation, C indicates a constant number represented by $n_a(n_r lc + k_a c)$. From this equation, this paper employs the Eq. (2) as the computational cost. Note that the computational cost of the Pittsburgh approach must be calculated as $p \cdot s(i)$ instead of $s(i)$, because the p number of LCSs solve the same problem at the same iteration.

4.4 Rule set design and task environment

The rule sets in OCS and the task environment are designed as follows.

- The condition part of CF (classifier) has the following four parts: (1) previous action (the 17 types described in the previous section); (2) a flag distinguishing whether a pentomino is overlapped or not (1 or 0); (3) a flag distinguishing whether a pentomino is totally enclosed by other pentominos or not (1 or 0); (4) a flag distinguishing whether a pentomino removes an overlapping area within a certain time or not (1 or 0). Furthermore, the action part of CF indicates primitive behaviors (17 types). According to this design, one example of 2 1 0 # 6 in CF shows that *if* `a previous action is the 2nd action & there is overlap & the pentominos are not enclosed`, *then* `act on the 6th action`. In this case, the mark of # indicates "don't care."
- An action is indicated by a discrete number, and the movement distance is also counted by a discrete number.

5 Analysis of OCS in Problem Scale

5.1 Experiment design

A simulation investigates the effect of the integration of the following mechanisms in problem scale.

- **Reinforcement learning mechanism:** Corresponds to individual single-loop learning.
- **Rule generation mechanism:** Corresponds to individual double-loop learning.
- **Rule exchange mechanism:** Corresponds to organizational single-loop learning.
- **Organizational knowledge reuse mechanism:** Corresponds to organizational double-loop learning.

Actually, the following four cases were tested with 24, 48, 96, 192 and 384 pentominos in the pentomino tiling problem. The reason for selecting the above numbers of pentominos is because the minimum area is known when the number of pentominos is a multiple of 12 as

shown in Fig. 6 (b). This indicates that the same type of pentominos in Fig. 6 (b) are used according to the multiple of 12. For example, two of the same types of pentominos are used in 24 pentominos.

- RGXK
- RGX (RGXK-K)
- RGK (RGXK-X)
- GXK (RGXK-R)

In the above cases, R, G, X and K indicate the mechanisms of **R**einforcement learning, rule **G**eneration, rule e**X**change, and Organizational **K**nowledge reuse, respectively. For example, RGXK indicates a case where all four mechanisms are included. Furthermore, RXK, which removes the G mechanism from RGXK, is omitted in this simulation. This is because the pentomino cannot create new rules needed to solve given problems.

5.2 Experimental setup

In the pentomino tiling problem, organizational knowledge and valuables in OCS are designed as follows.

- **Organizational knowledge** in this simulation is a set comprising each pentominos rule set that is acquired by 12 pentominos in advance and is reused as the initial rule sets of the pentominos. In detail, the y numbers of the pentominos utilize the rule sets as follows, where $RS_y(x)$ indicates the rule set of the x-th pentomino whose total number is y.

$$RS_y(x) \leftarrow RS_{12}(mod((x-1),12)+1),\ x=1,\cdots,y$$

- **Variables** in OCS are set as shown in Table 3.

5.3 Experimental results

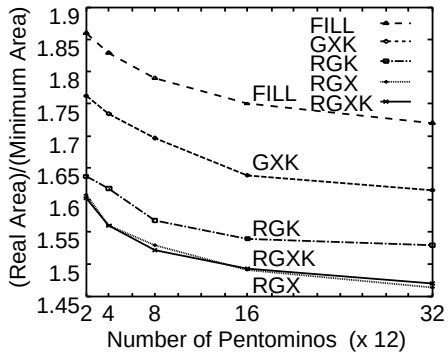
Figures 7 (a) and (b) show the results of *goodness* ((real area)/(minimum area)) and *computational cost* (sum of steps), respectively. These indexes are indicated in the vertical axes, and the number of pentominos, represented in a unit of a multiple of 12, is indicated in the horizontal axes. FILL indicates a result in which all pentominos are placed by minimizing the area in one try.

In the experiment, the results of FILL and others are averaged from 100 and 5 five situations with different random seeds, respectively. For the case of using the organizational knowledge reuse mechanism, the steps needed to acquire organizational knowledge are added to the results.

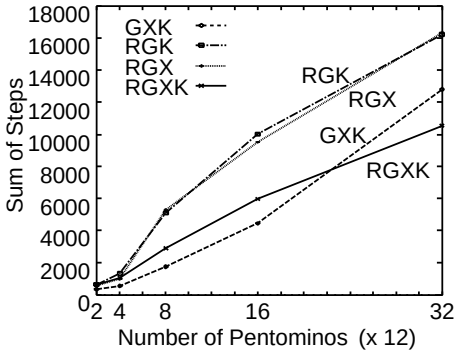
5.4 Discussion

From the results, the following implications are revealed:

- **RGXK and RGX in goodness:**
  As shown in Fig 7 (a), the goodness of RGXK and RGX is better than that of the others cases. This indicates that the integration of reinforcement learning, rule generation and rule exchange contributes to improving goodness. On the other hand, the goodness of both RGK and GXK is worse because (1) the pentominos in RGK cannot find other solutions due to a lack of rule exchange especially when the solution enters the local minimum; (2) the pentominos in GXK cannot learn to acquire their own appropriate functions due to a lack of learning process. Finally, the goodness of FILL is worst because each pentomino has only one chance to place itself.



(a) **Goodness: (Real area)/(Minimum area)**



(b) **Computational cost: Sum of steps**

Figure 7: **Goodness and computational cost**

- **RGXK in computational cost:**
  As shown in Fig 7 (b), the computational cost in GXK is the smallest when the number of pentominos is small, but that in RGXK becomes smallest as the number of pentominos increases. This is because the functions of the pentominos in GXK do not change drastically due to a lack of reinforcement learning. This makes the solution converge first. However, the number of pentominos who do not have rules matched to an environment increases as the number of pentominos increases. This makes the environment of multiagents change through the creation of new rules. Thus, the computational cost in GXK becomes large. RGXK, on the other hand, involves a change of the environment by reinforcement learning, but the environment does not often change by rule generation. This is because pentominos learn to reinforce necessary rules to avoid encountering an unknown environment. From this factor, RGXK shows the smallest computational cost as the number of pentominos increases.
- **Integration of the four learning mechanisms:**
  From the above analysis, we have found that the goodness or the computational cost becomes worse when one of the learning mechanisms is missing.

This factor suggests the limits to improve goodness and computational cost using only each mechanism itself. On the other hand, the integration of the four learning mechanisms contributes to finding good solutions with less computational cost, because each mechanism works in a different dimension and makes up for the defects of the other mechanisms. That is, the effective knowledge is utilized by *exploiting* the characteristics of a search space through the rule exchange and the organizational knowledge reuse mechanism. In addition, the ineffective knowledge is "not reinforced"/"removed" by *exploring* another search space through the reinforcement learning and the rule generation mechanisms. From the above discussion, the integration of the four learning mechanisms is sufficiently useful for practical and engineering domains because good results can be obtained as the problem size becomes larger.

6 Comparison with Conventional LCSs

6.1 Experiment design and setup

This section measures how OCS works by comparing it with two conventional LCSs. The following three cases are tested in the same pentomino tiling problem with 24 pentominos.

- Michigan Approach [Holland and Reitman, 1978]
- Pittsburgh Approach [Smith, 1983]
- OCS

The experimental setups of the Michigan approach, the Pittsburgh approach and OCS are shown in Table 3.

Table 3: **Variables in Michigan, Pittsburgh, and OCS**

	Michigan	Pittsburgh	OCS
Population size	1 LCS	24 LCSs	
FIRST_CF	30		
MAX_CF	50		
GA_STEP	20 steps	1 iteration	20 steps
Crossover	1 point crossover		rule exchange
GENERATION_GAP	0.1		
BORDER_ST	—		–50
Mutation	1 bit change		—
Mutation rate	0.05		—
Inversion rate	—	0.05	—
Credit assignment	profit sharing	—	profit sharing

* FIRST_CF, MAX_CF, GA_STEP, GENERATION_GAP, and BORDER_ST respectively indicate the number of initial rules of each pentomino, the maximum number of rules of each pentomino, the interval steps in GA operations, the ratio of operated rules, and the lowest strength of the rule not for removal.
* Note that (1) the parameters are decided through careful preliminary examinations in order to effectively show the essential results and (2) the tendency of the results does not change drastically with the parameter setting.

6.2 Experimental results

Table. 4 shows the results of *goodness*, *computational cost* and *vibration times* using the Michigan approach, the Pittsburgh approach and OCS, respectively. In particular, *vibration times* are counted when the size of the area is not converged.

Table 4: **Comparison of Michigan, Pittsburgh and OCS**

	Michigan	Pittsburgh	OCS
Goodness	1.87	1.57	1.60
Computational Cost	1039	16064	437
Vibration Times	3	0	0

In the experiment, all results are averaged from five situations with different random seeds. In particular, for the case of using the organizational knowledge reuse mechanism in OCS, the steps needed to acquire organizational knowledge are added to the results. Furthermore, the *computational cost* in the Pittsburgh approach is adjusted by multiplying p, which is the number of LCSs. This is because the same problem is solved by each LCS at the same iteration.

6.3 Discussion

From this table, the following implications are suggested.

- **Goodness and computational cost:**
  The computational cost in the Michigan approach is small, but the goodness itself is not good. The goodness in the Pittsburgh approach, on the other hand, is quite good, but a huge computational cost is required. Compared with these results, the goodness of OCS is slightly larger than that of the Pittsburgh approach, but the computational cost is the smallest. These are because some of learning mechanisms in OL are missing both in the Michigan and Pittsburgh approaches whereas OCS uses all four mechanisms.

  In detail, the Michigan approach includes individual single-loop learning (reinforcement learning) and individual double-loop learning (crossover and mutation)†, but the other two mechanisms are missing. The Pittsburgh approach, on the other hand, includes the individual single- & double-loop learning (inversion and mutation) and the organizational single-loop learning (crossover), but organizational double-loop learning is missing. Due to a lack of organizational single-loop learning in the Michigan approach, three mechanisms (individual single- & double-loop learning and organizational single-loop learning) cannot be integrated, even though exper-

†Crossover and mutation in the Michigan approach contribute to creating new rules but do not work at the organizational level.

imental results of section 5 found that this integration contributes to improving goodness. Therefore, the goodness is not improved in the Michigan approach. Furthermore, a lack of organizational double-loop learning in both approaches requires a much larger computational cost than OCS. In particular, the computational cost in the Pittsburgh approach is quite high, because many LCSs address to solve the same problem in spite of the fact that one LCS works in the Michigan approach and OCS approaches. From these discussions, OCS has found good solutions at lower computational cost.

- **Vibration times:**
  In general rule based systems, effective rules in the first several steps sometimes become unnecessary or ineffective at the end. This defect cannot be removed entirely from either the Michigan or the Pittsburgh approaches. In particular, the Michigan approach is more difficult than the Pittsburgh one to remove the rules that cause the above dilemma. This is because the former approach tries to remove such rules by genetic operation within its rule base, but the latter one can remove the rules in a unit of a rule base. This indicates an effectiveness of diversity (a large population) in the genetic operation. From this factor, the value of the area in the Michigan approach vibrates three times in five simulations, and this leads to bad goodness and an increase in computational cost. Compared with these features, the above dilemma does not happen so much in OCS because the dispersed LCSs reduce an influence towards the total performance even if some of the LCSs get into the above dilemma.

7 Related Work

As described in the previous sections, good solutions with less computational cost are attained by integrating the four learning mechanisms in OL. From this viewpoint, conventional multiagents and related research only focus on *some* parts of the concept of OL. Note that the below discussion does not include all methods of conventional research but just focuses on general methods.

- **Multiagent reinforcement learning** generally focuses on how to improve collective performance by utilizing individual knowledge prepared in advance. From the viewpoint of OL, only individual single-loop learning is the focus here.
- **Distributed problem solving in DAI** generally focuses on cooperation/coordination among agents. Some of research handle cases in which roles of agents change according to the situation or cases in which knowledge/information is exchanged among agents or through understanding other agents. From the viewpoint of OL, the former focuses on individual single- and double-loop learning, and the latter focuses on organizational single-loop learning.
- **Evolutionary computations** like genetic algorithms are considered to be a kind of problem solving method with many individuals. Based on this interpretation, individual double-loop learning (mutation) and organizational single-loop learning (crossover) are focused on from the viewpoint of OL.

What is important to mention here is that (1) it is necessary to improve each method in conventional research in order to make performance increase even though there are limitations to this improvement and (2) the integration of the four learning mechanisms in OL has the potential to overcome the limitations, even though each learning mechanism is a simple one, that is, each is not improved by specific or elaborate techniques. From this claim, this paper finds that new viewpoints such as OL are indispensable to derive the above two characteristics. In particular, if improvements of particular methods or techniques are fully concentrated to study in their domain, it is difficult to make clear what kinds of methods contribute to improving performance in multiagent environments.

From this fact, this paper arrives at the conclusion that new viewpoints are needed to develop new suggestions which cannot be found from the conventional viewpoints. Especially in this research, we have found that (1) the integration of the four learning mechanisms in OL is important factor for improving performance in multiagent environments, (2) this integration has great potential in overcoming the limitations of methods in conventional research.

8 Conclusion

This paper investigates the performance of OCS from the viewpoint of OL, compares it with conventional LCSs, and analyzes the characteristics of research on multiagents from the viewpoint of OL. The main results are summarized as follows: (1) the integration of the four learning mechanisms in OL is effective in finding solution and computational cost; (2) OCS finds good solutions at less computational cost than conventional LCSs; and (3) the concept of OL has the potential to overcome the limitations of conventional multiagent methods and techniques.

Future research will include the following.

- An analysis of other types or levels of learning mechanisms, and an investigation of integration effect.
- An investigation of the relationships between organizational structure and performance (both solutions and computational cost).

References

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

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

[Gasser and Bond, 1988] L. Gasser and A. Bond. *Readings in Distributed Artificial Intelligence*, Morgan Kaufman Publishers, 1988.

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

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

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

[Kim, 1993] D. Kim. "The Link between individual and organizational learning," *Sloan Management Review*, Fall, pp. 37–50, 1993.

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

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

[Smith, 1983] S.F. Smith. "Flexible learning of problem solving heuristics through adaptive search," *1983 International Joint Conference on Artificial Intelligence (IJCAI'83)*, pp. 422–425, 1983.

[Sutton and Bart, 1998] R.S. Sutton, A.G. Bart. *Reinforcement Learning – An Introduction –*, The MIT Press, 1998.

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

[Weiss and Sen, 1996] G. Weiss and S. Sen. *Adaption and Learning in Multi-Agent Systems*, Lecture Notes in Artificial Intelligence, Vol. 1042, Springer-Verlag, 1996.

[Weiss, 1997] G. Weiss. *Distributed Artificial Intelligence Meets Machine Learning – Learning in Multi-Agent Environments –*, Lecture Notes in Artificial Intelligence, Vol. 1221, Springer-Verlag, 1997.