How to Design Good Learning Agents in Organization?

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

Abstract

This paper categorizes four types of multiagent learning in terms of both goals and evaluations in agents, and investigates the characteristics of each categorization to find an effective type for designing learning agents. Since the characteristics in this categorization are affected by the learning mechanisms of agents, the characteristics are investigated by referring to organizational learning in organization and management science as one of methods. Through intensive simulations on a complex domain problem, the following implication has been revealed: agents that pursue their own goals and are evaluated according to their total results show high performance in comparison with other types of agents when all learning mechanisms in the organizational learning are employed.

Keywords: multiagent design, learning agents, organizational learning, learning classifier system

1 INTRODUCTION

What components or elements are needed to improve the performance in multiagent environments? How do we design good learning agents in dynamical environments? These are very important questions to clarify for both practical and engineering uses. To answer these questions, a lot of research on multiagent (Tan 1993, Weiss 1996, Weiss 1997) has been reported in recent years. Some examples include reinforcement learning (Sutton 1998), evolutionary computation (Goldberg 1989) [1] and distributed problem solving in distributed artificial intelligence (DAI) (Gasser 1988). However, such research seems to have concentrated on improvements in particular methods or techniques in multiagent environments.

Unlike these studies, our research analyzes what kinds of properties are embedded in environments or how these embedded properties affect the collective performance in order to investigate the ways of designing *good* learning agents. Although there are some design frameworks for communication in multiagent environments (Balch 1995) or for team behaviors (Collinot 1996), they do not directly support the design of *learning* parts of agents, especially in environments where all agents learn at the same time. To address this issue, our previous research focused on organizational learning (OL) (Argyris 1978, March 1991, Cohen 1995) in organization and management as one of methods and investigates the characteristics of OL to find essential components for designing learning agents (Takadama 1998c). However, this research did not discuss (1) a goal of agents and (2) an evaluation of agents' behaviors in detail, even though both of there points are essential in learning. As a result, this paper categorizes multiagent learning from the viewpoints of goal and evaluation, and finds an effective category for the design of learning agents considering the components based on OL.

This paper is organized as follows. Section 2 starts by categorizing multiagent learning, and Section 3 describes organizational learning and its model. An example for analyzing the characteristics of each category is shown in Section 4. Section 5 presents our simulations, and our conclusions are finally made in Section 6.

2 CATEGORIES IN MULTIAGENT LEARNING

In the categories of multiagent learning, Moriarty focused on the goal of agents and divided research on multiagent learning into two categories (Moriarty 1998): (1) agents pursue their own goals

*Paper submitted to the 1999 Genetic and Evolutionary Computation Conference (GECCO'99)

[1] This can be roughly considered as a kind of problem solving method with many individuals.

and (2) agents achieve shared goals. However, his research neither mentions which category is effective in multiagent learning nor focuses on an evaluation of agents' behaviors which is one of the indispensable points in the learning. To address this issue, this paper categorizes four types of multiagent learning in terms of goals and evaluations as shown in Fig. 1, and investigates the characteristics of each category.

- **Type 1:** Agents pursue their own goals and their behaviors are evaluated according to their own results.
- **Type 2:** Agents pursue their own goals and their behaviors are evaluated according to their total results.
- **Type 3:** Agents achieve shared goals and their behaviors are evaluated according to their own results.
- **Type 4:** Agents achieve shared goals and and their behaviors are evaluated according to their total results.

Goal \ Evaluation	Own result	Total result
Own	TYPE 1	TYPE 2
Shared	TYPE 3	TYPE 4

Figure 1: **Category in multiagent learning**

Note that the goal in multiagent environments is assumed to be shared when an agent cannot achieve its goal without cooperation among other agents, and it is not assumed to be shared when an agent can achieve its goal by itself even if all agents have the same goals. This claim indicates that (1) having the same goals must be distinguished from sharing goals, and (2) information on other agents is required to achieve shared goals through cooperation. To understand this claim, let us look at the pursuit problem as one example. In this example, predators (hunters) aim to track down one prey (animal), and all predators have the same goal. However, the goal is shared if more than one predator is needed to track down a prey, and the goal is not shared if one predator can track down a prey by itself. This implies that the former predator needs information on other predators (*e.g.*, location) to achieve shared goals but the latter predator, on the other hand, does not need the information on other predators.

3 ORGANIZATIONAL LEARNING AND ITS MODEL

3.1 ORGANIZATIONAL LEARNING

Organizational learning (OL) has been studied in the context of organization and management science. OL is roughly characterized as organizational activities for improving organizational performance, which cannot be achieved at an individual level. In particular, OL consists of the following four kinds of learning (Argyris 1978, Kim 1993):

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

This categorization stipulates that (1) there are individual and organization levels in the learning, and (2) each learning can be classified as a single or a double type. However, a *norm* in the above learning has not been defined clearly from a computational viewpoint. Thus, this paper assumes that (a) an *individual norm* and an *organizational norm* are respectively implemented by individual and organizational knowledge and (b) *individual knowledge* and *organizational knowledge* are respectively implemented by a rule set and a set of individual knowledge (rule sets). Note that the above assumptions only support to the reinterpretation of the four learning mechanisms from a computational viewpoint and do not discuss the detailed implementation. Based on this claim, this paper defines *computational organizational learning* as "learning that includes four kinds of reinterpreted loop learning".

3.2 ORGANIZATIONAL-LEARNING ORIENTED CLASSIFIER SYSTEM

3.2.1 Aim of agent and function

An Organizational-learning oriented Classifier System (OCS) (Takadama 1998a) is a GBML (Genetics-Based Machine Learning) architecture. OCS is composed of many Learning Classifier Systems (LCSs) (Goldberg 1989, Holland 1978), which are extended to introduce four kinds of reinterpreted loop learning mechanisms described in the previous section. In this model, agents are implemented by their own LCSs and divide given problems by acquiring their own appropriate *functions* through interaction among agents 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*, and a *function* is defined as a rule set (which is one component in LCS). 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 for acquiring 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 due to a change in the rule strength. Furthermore, an affection of learning in agents occurs when situations change to others, because a sequence of actions changes according to situations even though the function remains the same.

3.2.2 Architecture

As shown in Fig. 2, OCS is composed of many agents, and each agent has the same architecture that includes the following problem solver, memory, and four learning mechanisms reinterpreted from OL.

< **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 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 the 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 an internal state of an action of fired rules.
- **Rule sequence memory** stores a sequence of fired rules to evaluate them. This memory is cleared after the evaluation.

< **Mechanisms** >

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

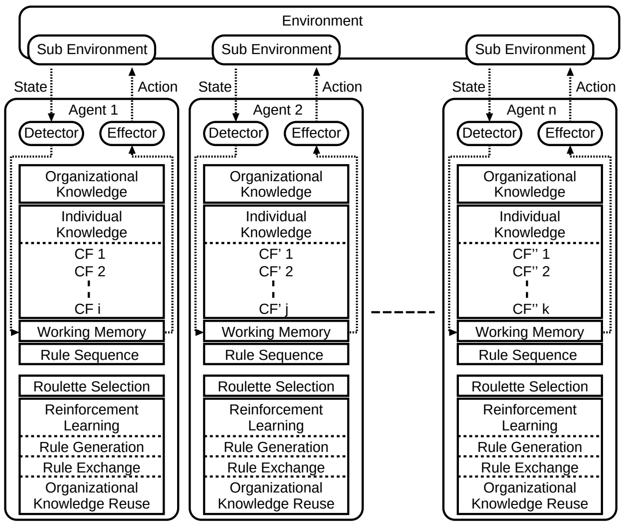

Figure 2: **OCS Architecture**

3.2.3 Learning in OCS

(1) Reinforcement learning mechanism: In OCS, the reinforcement learning (RL) mechanism enables agents to acquire their own appropriate actions which are is required to solve given problems. In particular, RL supports agents in learning 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 the performance within individual rules in computational organizational learning. In detail, this mechanism works as shown in Fig. 3, and is implemented by *profit sharing* (Grefenstette 1988), which reinforces a sequence of rules at once when agents obtain some rewards [1].

(2) 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. 3. 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

[1]Detail credit assignment in OCS was proposed in (Takadama 1998b)

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 for a replacement by new rules.

(3) Rule exchange mechanism: In OCS, agents exchange rules with other agents at a particular time interval (CROSSOVER_STEP [1]) to solve given problems that cannot be solved at an individual level as shown in Fig. 3. 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. 4, CFs are sorted in 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, the rules whose strength is higher than a particular value (BORDER_ST) are not replaced to avoid unnecessary crossover operations. The strength of replaced rules are reset to their initial values following this operation. This is because effective rules in some agents are not always effective for other agents in multiagent environments.

(4) Organizational knowledge reuse mechanism: Finally, agents in OCS store a set comprising each agent's rule set (individual knowledge) as the knowledge on the division of work when they solve given problems most effectively[2], and reuse it when REUSE_FLAG is TRUE [3]. For example, n number of agents most effectively solve problems with using their rule sets, a set comprising each agent's rule set is stored as shown in Fig. 5. 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.

[1]The step is defined in section 4.3.

[2]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 optimal most of the time but means the best since the agents have experimented up to the present.

[3]Although REUSE_FLAG can be set as TRUE in anytime, a reusing time in this stage of OCS is fixed at the iteration 0. In particular, the iteration is defined in section 4.3.

```
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, CROSSOVER_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 3: **Four learning mechanism algorithms**

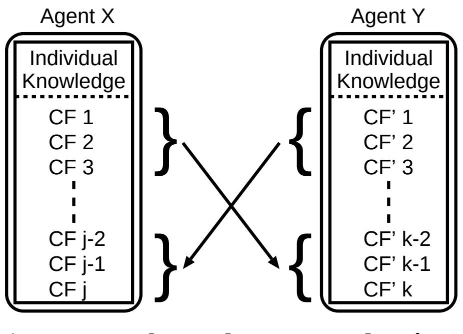

Figure 4: **Rule exchange mechanism**

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 the performance through the change of a set comprising each agent's individual rules itself in computational organizational learning. Agents in this stage of OCS cannot use organizational knowledge until this knowledge is divided into each individual knowledge. This indicates that agents cannot utilize 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 utilized in an unit of multiagent organization, while the latter knowledge is useful for all agents.

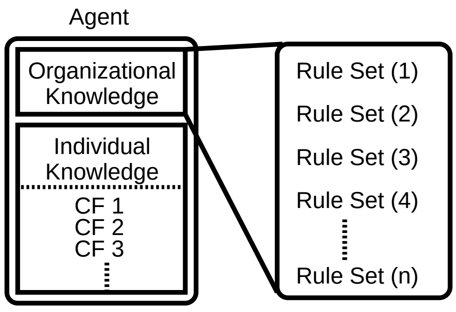

Figure 5: **Organizational knowledge reuse mechanism**

3.2.4 Supplemental Setup

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

3.3 COMPONENTS FOR DESIGNING LEARNING AGENTS

Using OCS, our previous research has found that the integration of four learning mechanism in OL is effective for both solutions and computational costs (Takadama 1998c), also found that the effectiveness of the integration is supported by the following three components: (1) different dimensions in learning mechanisms, (2) interaction among various levels and types of learning mechanisms in addition to interaction among agents, and (3) a combination of exploration at an individual level and exploitation at an organizational level. Three components are effective because they respectively (1) make up for the defects of the other single mechanism, (2) overcome a limitation of each mechanism in improvements of solutions and computational costs, and (3) *explore* an other search space by "not reinforcing"/"removing" ineffective knowledge through the individual loop learning mechanisms and *exploit* the characteristics of a search space by utilizing effective knowledge through the organizational loop learning mechanisms.

4 PENTOMINO TILING PROBLEM

4.1 PROBLEM DESCRIPTION

Pentomino is a figure that combines 5 squares as shown in Fig. 6 (a), and its tiling problem is to appropriately place pentominos with minimizing the area that encloses all pentominos without 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) which finds a parts layout that minimizes total wiring length (in this case, each part corresponds to each pentomino (Takadama 1998a)).

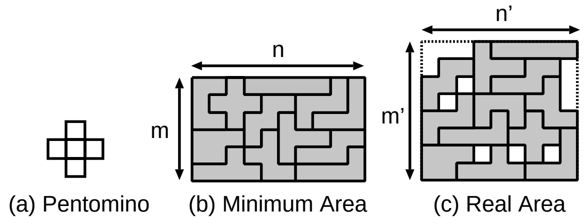

Figure 6: **Pentominos**

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 overlap. In detail, the pentominos have 17 primitive actions such as stay, move, or rotation.

As a concrete problem setting, all pentominos are initially placed at random without considering 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 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 which finds 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 $= \frac{Real\ area}{Minimum\ area}$
- Computational cost
  $= \sum_{i=1}^{iteration\_in\_convergence} step\ (i)$

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, "*i*", "*step* (*i*)", and "*iteration_in_convergence*" respectively indicate the iterations, 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.

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 contents: (1) previous action (17 types described in the previous section); (2) flag distinguishing whether a pentomino is overlapped or not (1 or 0); (3) flag distinguishing whether a pentomino is totally enclosed by other pentominos or not (1 or 0); (4) 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 represents that *if* `previous action is the 2nd action & overlap & not enclosed` *then* `act the 6th action`. In this case, the mark of # indicates "don't care".
- An action is indicated by a discrete number, and a movement distance is also counted by a discrete number.

5 SIMULATION

5.1 EXPERIMENT DESIGN

A simulation investigates the characteristics of four categories (type 1 $\sim$ 4) in multiagent learning environments by referring to components for the design of learning agents based on OL (Takadama 1998c). All four types are tested with 24 pentomino tiling problems in the following two cases. 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). Especially in 24 pentominos, there are two types of the same pentominos.

- Type 1 $\sim$ 4 with RGX (RGXK–K) mechanisms
- Type 1 $\sim$ 4 with RGXK mechanisms

In the above cases, R, G, X and K indicate the mechanisms of **R**einforcement learning, rule **G**eneration, rule e**X**change, and an organizational **K**nowledge reuse, respectively. For example, RGXK indicates the case when all four mechanisms are included.

5.2 EXPERIMENTAL SETUP

In the pentomino tiling problem, the goal and the evaluation in the four types of multiagent learning and the learning mechanisms in OCS are designed as follows.

- **Goal and evaluation** are set as follows: *Own goal* is to maximize the space that adjoins other pentominos, and *own result* is evaluated according to the total number of adjoining spaces. *Shared goal*, on the other hand, is to minimize the area that encloses all pentominos without overlap, and *total result* is evaluated according to the size of the area.
- **Organizational knowledge** in this simulation is a set comprising each pentomino's rule set acquired by 12 pentominos in advance and is reused as an initial rule set of 24 pentominos. In detail, 24 pentominos utilize rule sets as follows, where $RS_y(x)$ indicates the rule set of the x-th pentomino whose total number is y.

  $$RS_{24}(x) \leftarrow RS_{12}(mod((x-1),12)+1), x = 1, \cdots, 24$$

- **Variables** in OCS are set as follows: `FIRST_CF` (the number of initial rules) is 30, `MAX_CF` (the maximum number of rules) is 50, `CROSSOVER_STEP` (the interval steps for crossover operations) is 20, `GENERATION_GAP` (the percentage of operated rules) is 10%, and `BORDER_ST` (the lowest strength of the rule not for removal) is -50.0 [1].

5.3 EXPERIMENTAL RESULTS

Table 1 shows the results of the four types of multiagent learning, which are evaluated according to *goodness* ((real area)/(minimum area)) and *computational cost* (the accumulated steps). In particular, Tables 1 (a) and (b) respectively show the results for RGX and RGXK cases, and "—" indicates that the value

[1]Note that (1) these parameters are decided through careful preliminary examinations to effectively show the effect of each learning mechanism, and (2) the tendency in OCS does not change dynamically according to the parameter setting.

of goodness does not converge within 200 iterations [1]. All results are averaged from five situations with different random seeds. Especially in the case of using the organizational knowledge reuse mechanism, the steps needed to acquire organizational knowledge are added to the results. Furthermore, the steps in the case of the shared goals are multiplied by the number of pentominos for the computational cost. This is because information on the location of all other pentominos is required to accurately calculate the size of the current area that encloses all pentominos. Therefore, the additional steps for all other pentominos are needed in one step.

Table 1: **Comparison of types 1 $\sim$ 4 in multiagent learning**

	Type 1	Type 2	Type 3	Type 4
Goodness	—	1.68	—	1.57
Computational Cost	—	343	—	2376

(a) RGX

	Type 1	Type 2	Type 3	Type 4
Goodness	—	1.60	—	1.57
Computational Cost	—	437	—	4326

(b) RGXK

5.4 DISCUSSION

From the results of the four types of multiagent learning, the following implications are discussed:

- **Evaluation based on own result: Type 1 & 3**

In the case of RGX in Table 1 (a), the goodness value for in types 1 and 3 (which evaluate the actions of pentominos according to their own results) does not converge within 200 iterations. This is because different evaluations among pentominos are performed even if the same total result. This kind of evaluation avoids a concentration on finding one solution (divergence), because the same actions are not always selected due to different evaluations among pentominos.

In the case RGXK in Table 1 (b), on the other hand, there is a possibility to converge the goodness value for types 1 and 3 due to the utilization of the characteristics of a search space. However, the results shows the same tendency as compared with RGK. This indicates that the effect of evaluation at an individual level is stronger than that of organizational knowledge reuse at an organizational level.

[1]Since this paper aims to investigate how a solution and a computational cost are affected by four types in multiagent learning to find essential points for designing learning agents, a comparison of the results with other methods is out of the scope of this paper. This comparison, however, will be published elsewhere.

- **Evaluation based on total result: Type 2 & 4**

Unlike the results in types 1 and 3, the goodness value in types 2 and 4 (which evaluate the actions of pentominos according to their total results) converges in both case of RGX and RGXK as shown in Tables 1 (a) and (b). This is because the evaluations among pentomino is consistent.

Especially in RGX as shown in Table 1 (a), the goodness of type 4 is better than that of type 2 and the computational cost of type 2, on the contrary, is better than that of type 4. The former reason is because pentominos in type 2 cannot recognize the current size of the area that encloses all pentominos, while pentominos in type 4 can do it. The latter reason , on the other hand, is because pentominos in type 4 requires a lot of computational costs to acquire all information on the location of other pentominos, while pentominos in type 2 do not require this.

As shown in Table 1 (b), the goodness of RGXK in type 2 becomes small but that in type 4 does not improve as compared with those of RGX because of the following reasons: (1) pentominos in type 2 can utilize the characteristics of a search space which cannot be obtained with RGX mechanisms; (2) all information on the location of other pentominos is enough to improve the goodness (which means that the characteristics of a search space is supplemental). On the other hand, the computational cost of RGXK in both types 2 and 4 increase as compared with that of RGX. This is because the steps required to acquire organizational knowledge in the case of 12 pentominos are added. However, the increase degree of the computational cost in type 2 can be ignore as compared with that in type 4. This indicates the effectiveness of type 2 with RGXK mechanisms.

- **Effective type for designing learning agents**

In multiagent learning environments, the actions of some agents affect the learning of other agents, and this kind of interaction often changes the results. As a result, it is difficult to recognize which actions of agents contribute to an improvement of performance before given problems are solved. Although it is difficult for agents in type 2 to solve this problem, type 4 has the possibility of solving it. This is because type 4 can evaluate each action by using all information of the other agents. However, a lot of computational costs are required to implement such mechanisms, and this makes it difficult to apply agents in type 2 for practical and engineering uses.

In type 2, on the other hand, the goodness of this type with RGXK is slightly larger than that of type 4 but the computational cost of this type is much smaller than that of type 4. This result indicates that agents in type 2 with RGXK have found a good solution with less computational cost, and shows a possibility of solving the above problem of multiagent learning even if agents cannot recognize which actions contribute to an improvement of performance in each time. From this fact, we arrive at a conclusion for designing learning agents that suggests the design of agents in type 2 with introducing the four loop learning mechanism of OL (RGXK mechanisms). Although we find the same characteristics in the example of PCBs design problems in the CAD domain, we must distinguish the kinds of problems or task domains in which agents in each type perform well (This means finding a good solution with less computational cost) and must also perform a lot of experiments with other examples to propose a guideline for designing learning agents.

6 CONCLUSION

This paper has introduced a new category in multiagent learning which is divided according to agents' goals and evaluations, and also has found an effective category for designing learning agents which suggests the design of agents in type 2 with introducing four loop learning mechanism of OL (RGXK mechanisms). The main results are summarized as follows: agents that pursue their own goals and are evaluated according to their total results show high performance in comparison with other types of agents when all learning mechanisms in the organizational learning are employed.

Future research will include the following.

- Many experiments with other examples to propose guidelines for the design of learning agents.
- An investigation of the characteristics of other types of multiagent learning and the effectiveness of introducing other concepts.

References

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

T.R. Balch and R.C. Arkin (1995). "Communication in reactive multiagent robotic systems", *Autonomous Robots*, Vol. 1, No. 1, pp. 27–52.

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

A. Collinot, A. Drogoul, and P. Benhamou (1996). "Agent Oriented Design of a Soccer Robot Team ", *The Second International Conference on Multi-Agent Systems (ICMAS'96)*, pp. 41–47.

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

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.

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

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

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

D.E. Moriarty, S. Handley, and P. Langley (1998). "Learning Distributed Strategies for Traffic Control", *The Fifth International Conference of the Simulation for Adaptive Behavior (SAB'98)*, pp. 437–446.

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

R.S. Sutton, A.G. Bart (1998). *Reinforcement Learning – An Introduction –*, The MIT Press.

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

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

K. Takadama, T. Terano, K. Shimohara, K. Hori and S. Nakasuka (1998c). "Making Organizational Learning Operational: Implication from Learning Classifier System", *ATR Technical Report, TR-H-257.*

M. Tan (1993). "*Multi-agent Reinforcement learning: Independent vs. Cooperative Agent*", The 10th International Conference on Machine Learning (ICML'93), pp. 330-337.

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

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