AH?%3X=,;X8~7?J,N`;R%7%9%F%`H=$N1~MQ *

Organizational-Learning Oriented Classifier System for Intelligent Multiagent Systems *

○ **Keiki Takadama**
Graduate School of Engineering
The University of Tokyo
keiki@ai.rcast.u-tokyo.ac.jp

Shinichi Nakasuka
Research Center for Advanced Science and Technology
The University of Tokyo
nakasuka@space.t.u-tokyo.ac.jp

Takao Terano
Graduate School of Systems Management
The University of Tsukuba
terano@gssm.otsuka.tsukuba.ac.jp

Abstract

Organizational-learning oriented Classifier System (OCS) is a new multiagent-based learning model with an evolutionary computational mechanism. It is composed of a production system, a reinforcement learning mechanism, and rule generation/exchange mechanisms together with introducing concepts of organizational learning. In OCS, agents acquire their own appropriate functions through interaction among their neighborhoods, and form an organizational structure without an explicit control mechanism. To investigate the effectiveness of our model, we have conducted intensive experiments on real scale printed circuit boards (PCBs) redesign problems in the computer aided design (CAD). The experimental results have suggested that (1) Total wiring length with OCS is shorter than that with human experts; (2) OCS finds feasible solutions even if the size of PCBs becomes large or the type of PCBs is different; and (3) Knowledge acquired with the small size of PCBs in OCS is also applicable to the large sized one, and this utilization of the knowledge contributes to reducing the iteration counts.

Topic Areas: organization and social structure, agent models and architectures

Keywords: organizational learning, learning classifier system, cooperative agents, multiagent reinforcement learning, print circuit board design

1 Introduction

Learning mechanisms in multiagent systems in the literature mainly have focused on reinforcement learning [14]. However, of course, these mechanisms need more sophisticated techniques; for example, adaptation to the environments and learning by communication and understanding are also important issues. To overcome such issues, we have proposed a novel evolutionary computational model: Organizational-learning oriented Classifier System (OCS) [10, 11, 13] which intrinsically has the following three components: (1) reinforcement learning, which is used in the same way as in the Distributed Artificial Intelligence (DAI) literature, (2) rule generation, which enables agents to adapt to dynamical environments by GA operations, and (3) rule exchange, which improves the organizational performance by communicating and understanding. OCS enables agents to acquire their own appropriate functions through the local interaction among them and form organizational structure which improves organizational performance. Furthermore, agents in OCS behave only according to their own local evaluation functions, instead of a global evaluation function as an explicit control mechanism.

In order to investigate the effectiveness of our model OCS, this paper applies OCS to one of the most complex, practical and engineering problems: Printed Circuit Boards (PCBs) re-design in Computer Aided Design (CAD) domain. Especially, this paper focuses on the following two issues: (1) the effectiveness of OCS in the different size or type of PCBs and (2) the utilization of the knowledge acquired in OCS.

This paper is organized as follows. Section 2 starts to explain concept of the organizational learning, and Section 3 gives the details of the organizational learning model using learning classifier systems. An example of an automated parts placement on PCBs is given in the section 4, and Section 5 discusses the experimental results in the PCBs re-design problems. Finally, the conclusion is given in Section 6.

2 Organizational Learning

2.1 Definition of Organizational Learning

The research on organizational learning has been developed in the context of management organizational sciences, and a lot of researches focus on economical market systems or human organizations [1, 2, 3, 8]. In management organizational sciences, organizational learning is generally defined as follows: "*Organizational*

* Paper submitted to the Third International Conference on Multi Agent Systems (ICMAS'98)

Learning is the creating, acquiring and transferring of *distinctions* and *practices* in the organizations. It is effective if it increases the organization's fit to its environment. Organizational learning implies behavior modification, including changes in relationships, in order to create the conditions for creating, acquiring and transferring distinctions and practices [3]".

In organizational learning, there are two kinds of research streams: a top-down approach and a bottom-up approach. The former aims to analyze the phenomena in one organization, and seeks for the methodology to improve the organizational performances. For example, these researches focus on how effectively middle managers assign the jobs to their men in a certain static organizational structure. The latter aims to analyze the phenomena caused by the activities of other organizations, and studies the effects of organizational structures in organizational problem solving. This research discusses what parts of the functions in the enterprises should be kept in the organization and what parts should be outsourced in order to progress their organizational activities interdependently. In this paper, we focus on the latter stream of organizational learning, and apply it to practical engineering problems.

2.2 Organizational Structure and Functions

In our organizational learning model, two technical terms are defined as follows.

- **Organizational Structure:** A structure in which agents acquire their own functions.
- **Function:** A sequence of behaviors determined by rules.
  - The rules are generated/deleted when they are useful/useless.
  - Agents learn adaptive strategies as the order of using the rules.

Since the organizational structure dynamically changes, agents must adapt to a new organization to maintain or improve the organizational performance. For effective adaptation to the new organization, agents perform by using the following two methods. First, agents generate and delete behaviors: This method enables agents to acquire new behaviors required to improve the organizational performance. Second, agents learn adaptive strategies to acquire the effective long sequence of behaviors: This method enables agents to cooperate with other agents. The long sequence of behaviors produces a lot of variety of behaviors, and this variety provides a chance to improve the total organizational performance.

3 Organizational-learning oriented Classifier System

3.1 OCS Architecture

In order to propose the organizational learning model, we employ Learning Classifier System (LCS) [7] [5] which enables agents to perform effectively without identifying an environment. LCS is a machine learning system which learns syntactically simple string rules to guide its performance in an arbitrary environment, and it is composed of a production system, a reinforcement learning mechanism, and a rule generation by Genetic Algorithms [5]. There are two types of approaches for the LCS: Michigan approach and Pittsburgh approach. Michigan approach is composed of one LCS and evaluates each rule in LCS with a credit assignment mechanism. Pittsburgh approach, on the other hand, is composed of a lot of LCSs and evaluates each LCS with selecting the elite LCSs. However, both approaches cannot be directly applied to multiagent learning systems because of the following defects.

- **Defects of Michigan Approach:**
  - It is difficult for agents to acquire different functions to cooperate each other in given tasks, because all agents access to the same LCS.
  - There are no rule exchange mechanisms among agents.
  - Since new rules are generated by recombining two parent rules, new generated rules are not always useful.
- **Defects of Pittsburgh Approach:**
  - A global objective function is required to select the elite LCSs.
  - Since the rules in the elite LCSs are exchanged in the crossover operation, most agents have a tendency to acquire the same rules. This crossover operation prevents agents from acquiring different functions to cooperate each other in given tasks.
  - Although rules can be exchanged between LCSs, new rules are not created when a new environmental state is recognized.
  - Since the same task is provided to a population of LCSs, this approach is not fit to a multiagent learning system but to a parallel learning system. Multiagent learning systems must solve given tasks by cooperating among agents.

To overcome the above defects, we have proposed Organizational-learning oriented Classifier System (OCS) [10, 11, 13] which is an extended multiagent version of LCS together with the concept of organizational learning as follows.

- As shown in Fig. 1, each agent does not employ the explicit global evaluation function, but employs the local evaluation function which is composed of the following four parts: (1) Rule base composed of a lot of rules (CFs: Classifiers), (2) Working Memory (WM) for storing information on the environmental state, (3) Memory for storing the rule sequence which is cleared after the rewards are distributed to agents, and (4) Evaluation Mechanism to evaluate the rule sequence.

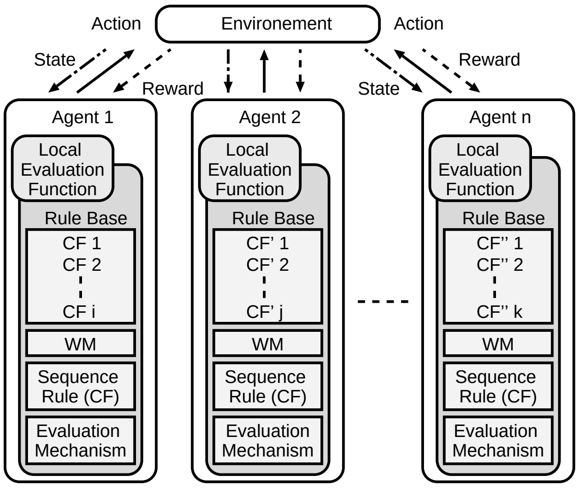


Figure 1: **Architecture of OCS**

- Each agent obtains its environmental state and determines the behavior according to the state. When the rewards are distributed to agents, each agent modifies or acquires its own function by self-organizing its own rule base.

- The rules are created when an agent encounters a new environmental state. When the number of the rules is more than `MAX_CF` which defines the max number of the rules, the lowest strength rule is removed and a new rule is generated. This method enables agents to adapt to the change of the environments. In the rule generation, the condition part is created to reflect the current situation, and the action part is determined at random. The strength of the rule is set to the same initial value.

- In the crossover operation for OCS, the ineffective rules are replaced with the effective rules between two arbitrary agents, instead of creating the large population with elite agents like the conventional evolutionary approach. Thus, this method reduces the computational costs. The `CROSSOVER_TIME`, `CROSSOVER_NUM` and `BORDER_ST` are respectively defined as the interval steps for crossover operations, the number of the replaced rules, and the rule strength which decides whether the rule must be replaced or not.

  – Every `CROSSOVER_TIME` step, two agents are selected at random, and the `CROSSOVER_NUM` rules are replaced. For example, when the agent X and Y are selected as shown in Fig. 2, the rules whose strength is low in the agent X and Y are replaced respectively with the rules whose strength is high in the agent Y and X. This method not only prevents getting into deadlock situations, but also improves organizational performance by propagating effective rules among agents.

  – In order to avoid the unnecessary crossover, the rule is not replaced when its strength value is higher than the `BORDER_ST`. This method contributes to quickly forming an effective organizational structure which improves organizational performance.

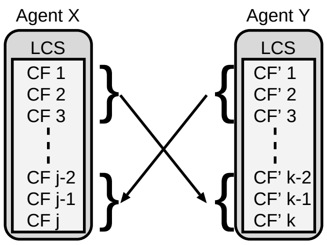


Figure 2: **Crossover Operation**

In OCS, we employ the following mechanisms and assumptions. `FIRST_CF` is defined as the number of rules which are provided to each agent at the beginning.

- A rule is selected according to a roulette selection used in usual LCS.

- At the beginning, the `FIRST_CF` rules in each agent are generated at random, and each strength value of rules is set to the same value.

- Since it is difficult to perfectly prepare the effective and complex rules beforehand, OCS employs the simple and rational rules and aims to generate the effective and complex rules from the simple and rational ones.

3.2 Multiagent Reinforcement Learning

In order to make agents cooperate with each other to complete given tasks effectively, we have proposed the multiagent reinforcement learning as shown in Fig. 3 †. This reinforcement learning is based on Profit Sharing [4] and is performed as an evaluation mechanism shown in Fig. 1. In this reinforcement learning, the positive rewards are distributed to all performed rules according to the geometrical increasing functions as shown in Fig. 3 (a) or Eq. (1), and the negative rewards are distributed to the rules as shown in Fig. 3 (b) or Eq. (2). This idea indicates that the first several steps are very important for the cooperation in multiagents. That is to say, a lot of local minimum solutions are mostly decided by the first several selected rules, because the behaviors of other agents are strongly affected by these rules.

Furthermore, in order to converge with finding the solution which is close to the best one, our method does not reinforce the rules in the case of $Best - (Best - Worst)/4 < x \leq Best$ as shown in Fig. 3 (c), where x is the value of the solution. In this equation, $Best$ / $Worst$ indicates the best/worst solution, and both solutions are updated in the process of solving the same given tasks continually. Moreover, the positive rewards are distributed in the case of $(Best + Worst)/2 \leq x \leq Best - (Best - Worst)/4$ or $Best < x$, and the negative rewards are distributed in the case of $x < (Best + Worst)/2$. Both positive and negative rewards contribute to extending the search space effectively.

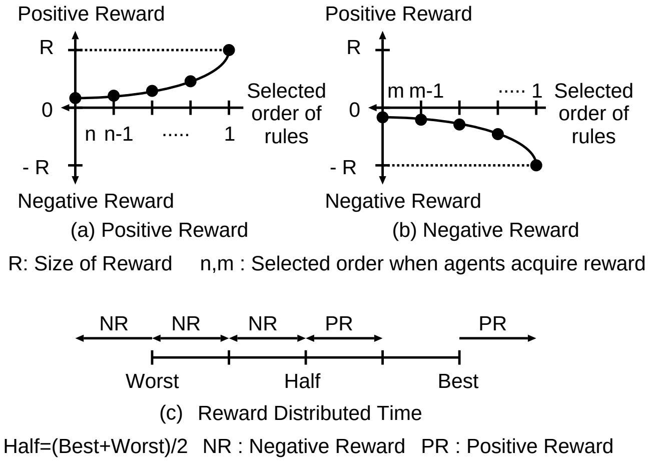


Figure 3: **Multiagent Reinforcement learning**

- **Positive Reward:**
$$ST(i) = ST(i) + R \cdot (\frac{1}{2})^{i-1}, where\ i = n, n-1, \cdots, 1 \quad (1)$$
- **Negative Reward:**
$$ST(i) = ST(i) - R \cdot (\frac{1}{2})^{i-1}, where\ i = m, m-1, \cdots, 1 \quad (2)$$

†The details will be published elsewhere [12]

In the above equations, ST represents the strength of CF, i represents the selected order of CF, n and m represent the max number of the selected CFs, and R represents the max size of the rewards. In the case that i is small, CF(i) is selected in the first several selections.

4 PCBs Design

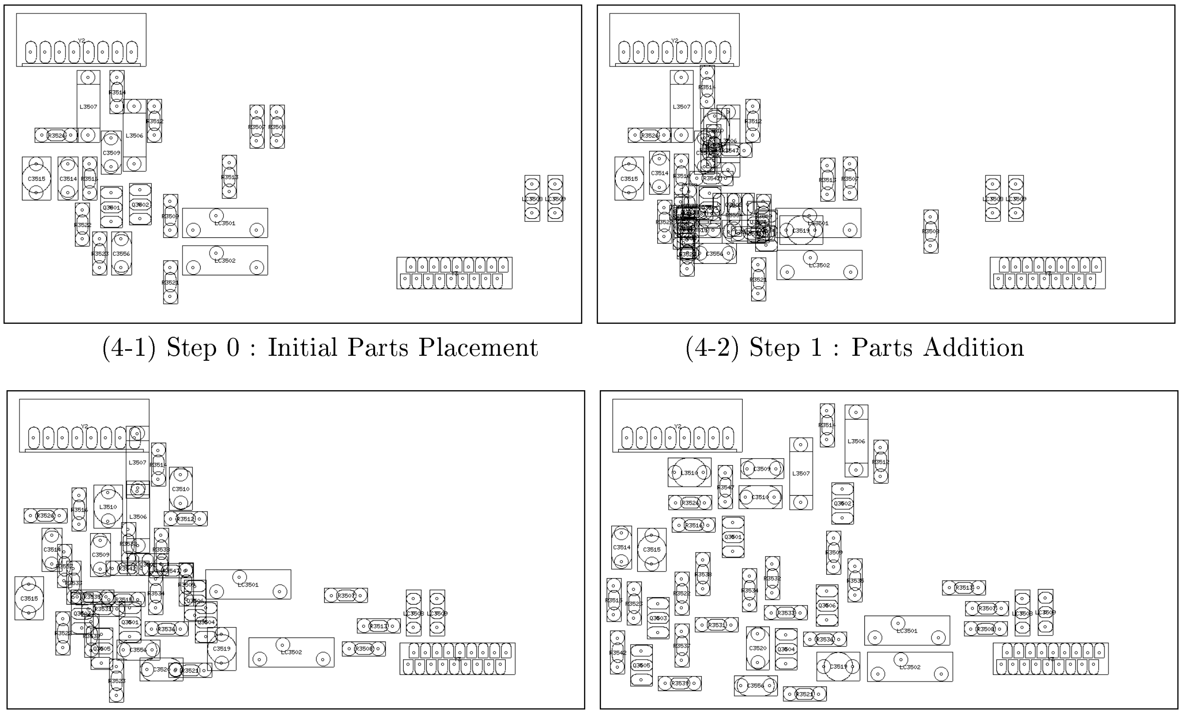
4.1 Problem Description

In the printed circuit boards (PCBs) design on CAD problems, the automated techniques for parts placement are required to reduce design periods and/or production costs. Especially in the PCBs re-design such as modification, fixing, and extension of the original parts layout, the problem is difficult because (1) the original parts layout should be remained as much as possible, (2) the severe constraints are added to the original design when new parts are placed, (3) it is impossible to clearly define an explicit evaluation function in advance, and (4) heuristic knowledge is not enough. Although various methods such as the knowledge-base system [15], Simulated Annealing [9] [6] or Genetic Algorithms [16] are proposed in the literature, the conventional methods have not been able to place effectively the parts as done by human experts. Furthermore, the supports of human experts are intrinsically required to satisfy the constraints and to optimize the layout of parts placement.

To validate the effectiveness of our model OCS, this paper applies it to the PCBs re-design problems as one of the real problems for practical and engineering use, and evaluates the total wiring length with using real CAD data. Furthermore, we focus on the effectiveness of OCS in the different size or type of PCBs and the utilization of the knowledge acquired in OCS. In this task domain, the parts are intelligent agents with their own LCSs, and acquire their own functions to minimize the total wiring length without an explicit control mechanism.

4.2 Part

In the parts addition re-design, new parts are added to the original parts as shown in Fig. 4–1. In order to make the total wiring length short, these parts are placed on the centroid location of the connected pins. Therefore, new parts overlap with the original parts as shown in Fig. 4–2. As an assumption, the parts can recognize these overlapping areas, but they do not know the total wiring length and their effective behaviors beforehand. In such a situation, each part moves to reduce the overlapping areas of other parts or rotation as shown in Fig. 4–3. Finally, all parts are placed as shown in Fig. 4–4.



(4-1) Step 0 : Initial Parts Placement (4-2) Step 1 : Parts Addition

(4-3) Step 10 : Parts Movement (4-4) Final Step : All Parts Placement

Figure 4: **Snapshot of Parts Placement**

Each part has some primitive actions: reducing overlapping areas of other parts, rotation, and so on. Using these primitive actions, the parts behave according to the rules in their own local evaluation functions as shown in Fig. 1. Rewards are distributed to the parts when all parts are placed, and each rule sequence of each part is reinforced according to the multiagent reinforcement learning method as shown in Fig. 3. In this case, the reward type such as negative or positive is decided by the value of total wiring length. Due to this reward, the functions of the parts are changed, and the organizational structure is also changed according to the functions.

5 Experiments

5.1 Method

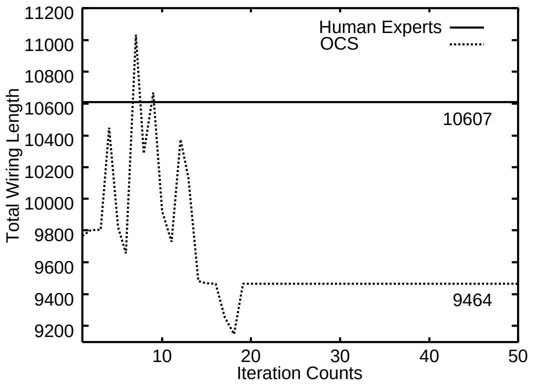
In the experiments, the following three cases are performed.

1. **Case 1: PCB1 (45 Parts)**
   20 parts are added to the 25 original parts which are already set on PCB and are used as an appliance product.
2. **Case 2: PCB2 (92 Parts) without priori knowledge**
   40 parts are added to the 52 original parts. The size of the PCB2 is bigger than that of the PCB1, and the types of both two PCB1 and PCB2 are different.
3. **Case 3: PCB2 (92 Parts) with knowledge acquired in PCB1**
   40 parts are added to the 52 original parts with utilizing the knowledge acquired in the PCB1.

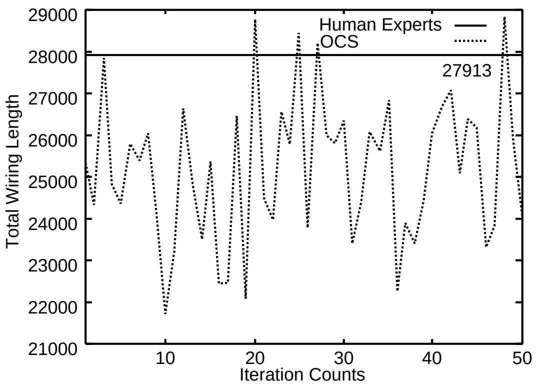
In each case, the additional parts overlap with the original parts at first. Second, the parts gradually move to the positions to satisfy their own constraints as shown in Fig. 4. Third, the parts repeatedly learn to acquire their own effective functions from the same initial placement when all parts are placed. This functions acquired in the cycle of learning contributes to finding appropriate placements with reducing the overlapping areas of the parts and minimizing the total wiring length. As a definition, we count one step when all parts perform one action, and count one iteration when the parts are all placed. The variables are set as follows: `FIRST_CF` is 50, `MAX_CF` is 100, `CROSSOVER_TIME` is 100, `CROSSOVER_NUM` is 5, `BORDER_ST` is -50.0, and R is 1.

5.2 Results

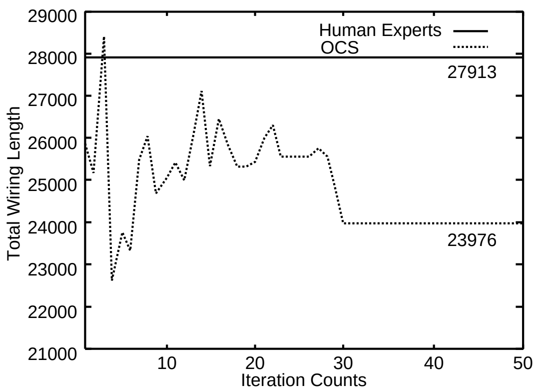
Fig. 5 shows a comparison of the total wiring length by human experts and OCS. In this figure, the x-axis indicates the number of iterations and the y-axis indicates the total wiring length. In the case 1, the total wiring length of OCS (9464) is converged and it is shorter than



(5-1) PCB1



(5-2) PCB2 without priori knowledge



(5-3) PCB2 with knowledge

Figure 5: **Iteration versus Total Wiring Length by Human Experts and OCS**

that of human experts (10607) as shown in Fig. 5–1. Since the size of PCB1 is small, the length is converged about 20 iterations. In the next case 2, OCS without priori knowledge finds feasible solutions even if the size of PCBs becomes larger than that of PCB1 and the type of PCB2 is different from PCB1. Although almost of the length in the case 2 are shorter that of human experts (27913) and some of them are shorter than those in the case 3, the length in the case 2 is not converged until 50 iterations as shown in Fig. 5–2. In this case, the length is converged after 62 iterations. Finally, in the case of the PCB2 with the knowledge acquired in the PCB1, the total wiring length of OCS (23976) is converged and it is shorter than that of human experts (27913) as shown in Fig. 5–3. By utilizing the knowledge for different type and size of PCB, the length is converged about 30 iterations.

In each case, the total wiring length of human experts is not improved even if the iteration counts increase. This is because it takes a half day for human experts to optimize the layout of parts placement, and there is no more time for optimizing a lot of PCBs re-design. On the other hand, it takes only 1 or 2 minutes for OCS to find one feasible solution when OCS is programmed in personal computers.

5.3 Findings and Discussions

From the experimental results in the automated parts placement on PCBs, the following findings are obtained:

- **Effectiveness of OCS:**
  Although PCBs re-design problems must satisfy a lot of constrains, OCS enables agents to find an appropriate placement without an explicit control mechanism. As a result, the total wiring length of OCS is shorter than that of human experts, even if appropriate behavior rules are not prepared beforehand. These results suggest that OCS is effective for the following problems: (1) the problem which is difficult to design an appropriate global evaluation function, (2) the problem which is difficult to prepare appropriate behavior rules beforehand, (3) the problem which must satisfy a lot of constrains, and (4) the domain where heuristic knowledge is not enough.

- **Scalability in OCS:**
  In OCS, the required time for each agent to perform in one step is a constant, even if the number

of agents increases. This is because (1) the agents in OCS obtain only local environmental states, instead of obtaining total environmental state, and (2) the rule exchange mechanism is performed only between agents. This suggests that OCS is scalable and can be applied any size of PCBs design problems.

- **Different Size/Type of PCBs in OCS:**
  Since OCS finds the feasible solutions even if the size and the type of PCB are different, OCS is effective for finding an appropriate placement in PCBs re-design problems. Especially, since almost of the length in OCS are shorter that of human experts, OCS performs well in any size or type of PCBs design problems. Although the convergence steps for minimizing the total wiring length in PCB2 is longer than that in PCB1, this result is not so big problems because OCS is scalable.

- **Utilization of Knowledge in OCS:**
  OCS with the knowledge acquired in other PCBs not only minimizes the total wiring length more than human experts, but also reduces the iteration counts. This results suggest that (1) OCS explores the search space which is extended in the first few iterations, and (2) OCS exploits the knowledge to reduce the overlapping areas as quickly as possible in the process of solving the same given tasks. Especially in the first few iterations, since the agents cannot utilize the knowledge effectively, they learn how to utilize the knowledge with extending search space. From these factors, OCS effectively addresses the trade-off between exploration and exploitation, and finds the short total wiring length with reducing iteration counts.

  Furthermore, the convergence steps/iterations do not become bigger in proportion to the size of PCBs. For example, the convergence iterations in PCB2 with the knowledge are not twice counts in PCB1 as shown in Fig. 5–1 and 5–3, even if the size of PCB2 with the knowledge is twice larger than the size of PCB1. From these results, the knowledge acquired in some PCBs is applicable to other PCBs, and this utilization of the knowledge contributes to reducing the iterations/steps to find the feasible solutions, even if the size of PCB becomes large. This results also indicates that the acquired knowledge is useful for practical and engineering use.

6 Conclusion

This paper has described a novel evolutionary computational model: Organizational-learning oriented Classifier System, and has shown its effectiveness with the PCBs re-design problems in CAD domain. OCS enables agents to acquire their own functions to minimize the total wiring length, and finds feasible solutions without an explicit control mechanism. Furthermore, our model is effective for the problems which are difficult to design an appropriate global evaluation function, or which are difficult to prepare appropriate behavior rules beforehand. OCS also performs effectively for the problems which must satisfy a lot of constrains like the PCBs re-design or the domain where heuristic knowledge is not enough. From the experimental results, we have found that (1) Total wiring length with OCS is shorter than that with human experts, (2) OCS enables agents to find an appropriate placement even if the size of PCBs becomes large or the type of PCBs is different, and (3) Knowledge acquired with the small size of PCBs in OCS is also applicable to the larger sized one, and this utilization of the knowledge contributes to reducing the iteration counts. Future researches include:

- Development of algorithms for multi-modal landscapes problems in multiagents system.
- Theoretical analysis in the our model OCS.
- Verification of the effectiveness of OCS in other domain problems.
- Verification of the effectiveness of OCS in the group reconstruction not only when new parts are added, but also when some parts are removed;
- Proposition of new mechanisms for OCS to solve the problems in the required time limitation;

Acknowledgment

This research is supported in part by "Information-Technology Promotion Agency", Japan, for Advanced Software Enrichment, and it also is supported in part by "System Theory of Function Emergence under the Grant-in-Aid for Scientific Research on Priority Area" of the Ministry of Education, Science, Sports, and Culture of Japan.

References

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

[2] R. Duncan and A. Weiss: "*Organizational Learning:Implications for organizational design*", in B. M. Staw (Ed.), Research in organizational behavior, Vol. 1, JAI Press, pp. 75-123, 1979.

[3] R. Espejo, W. Schuhmann, M. Schwaninger and U. Bilello: "*Organizational Transformation and Learning*", John Wiley & Sons, 1996.

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

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

[6] M. Hirahara, N. Oka and K. Yoshida:, "*Automatic placement using static and dynamic groupings*", Engineering Design & Automation, Vol. 3, No. 2, 167-178, 1997.

[7] J. H. Holland: "*Properties of the Bucket Brigade Algorithm*", ICGA '85, pp. 1-7, 1985.

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

[9] C. Sechen and A. Sangiovanni-Vincentelli: "*The TimberWolf placement and routing package*" IEEE Journal Solid-State Circuits, SC-20,2, pp. 510-522, 1985.

[10] K. Takadama, K. Hajiri, T. Nomura, M. Okada and K. Shimohara: "*Learning Model for Adaptive Behaviors as an Organized Group of Swarm Robots*", The 2nd International Symposium on Artificial Life and Robotics (AROB'97), pp. 103-106, 1997.

[11] K. Takadama and T. Terano: "*Good Solutions will Emerge without a Global Objective Function: Applying Organizational Classifier System to Printed Circuit Board Design*", IEEE 1997 International Conference on Systems, Man, and Cybernetics (SMC'97), pp. 3355–3360, 1997.

[12] 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), 1998, to appear.

[13] K. Takadama, S. Nakasuka and T. Terano: "*Printed Circuit Board Design via Organizational–Learning Agents*", Applied Intelligence: Special Issue on Intelligent Adaptive Agents, 1998, to appear.

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

[15] H. Yoshimura: "*Knowledge-based placement and routing system for printed circuit board*", PRICAI '90, pp. 116-121, 1990.

[16] T. Yoshikawa, T. Furuhashi and Y. Uchikawa: "*Coding Methods for Automatic Placement of Parts on Printed Circuit Boards*" (in Japanese), Trans. of The Institute of Electrical Engineers of Japan, Vol. 115-D, No. 5, pp. 642-651, 1995.