Construction of Players' Action for Robocup Soccer Using Graph Structured Program Evolution

Yasuaki Horima, Shinichi Shirakawa, Noriko Yata, and Tomoharu Nagao
Graduate School of Environment
and Information Sciences
Yokohama National University
Yokohama, Japan
Email: horima@nlab.sogo1.ynu.ac.jp

***Abstract*—In recent years, many researchers addressed multi-agent system. Multi-agent system is the system consisted of multiple robots that have only limited capability. Robocup simulated soccer is proposed as a test bed of multi-agent system. It has a subtask called keepaway soccer. Automatic construction of the strategy of multi-agent system is required because it is difficult. Therefore, we purpose construction of the strategy for multi-agent system by graph structured program evolution (GRAPE) in keepaway soccer. GRAPE is the method of construction of graph-structured programs automatically.**

***Index Terms*—Multi-agent system, Robocup soccer, Keepaway soccer.**

I. Introduction

Recently, autonomous robots have been put to practical use in our lives. In the future, multiple autonomous robots will run in our environment. When multiple robots run, cooperation of multiple robots is important. It is called multi-agent system that the system working by multiple agents such as autonomous robots. Cooperative action enables robots to work more efficiently and to do tasks that are too hard to do by a single robot. However, it is difficult to construct the cooperative action in an environment with multiple robots. Therefore, it is required the method for automatic construction of the cooperative action of robots.

Many researchers have acquired strategies of agents by machine learning (e.g. genetic programming (GP)[15], reinforcement learning (RL)[16]). GP constructs tree structured strategies. Tree structure cannot construct algorithms of reusing nodes but graph structure can do that. Therefore, it is proposed that methods automatically construct graph-structured algorithm (e.g. GRAph structured Program Evolution (GRAPE)[5][6], genetic network programming (GNP)[8]).

Robocup simulated soccer is proposed as a test bed for machine learning of multi-agent system[1][2]. This presents the environment with simulated real elements. Soccer players on the field are multiple agents. Agents using noisy and limited data play soccer in the simulated field. Actions of agents and movement of the ball are uncertain and unpredictable. Agents must cooperate in these environments. In this paper, we aim to construct soccer actions of agents by using GRAPE in robocup soccer.

II. Related works

A. Keepaway soccer

Robocup simulated soccer has a subtask called keepaway soccer. This is a task that robocup simulated soccer is simplified and a test bed for machine learning[3][4]. In keepaway soccer, keepers try to keep possession of the ball for as long as possible. On the other hand, takers try to get possession of the ball. Agents play in the limited rectangular area. Figure 1 shows overview of keepaway soccer.

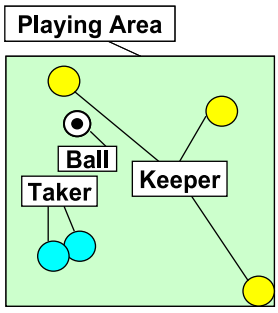


Fig. 1. Overview of keepaway soccer.

B. Methods of constructing graph-structured algorithms

Various methods of automatic construction of graph-structured algorithms was proposed such as GRAPE, GNP and Genetic Automata Generation(GAUGE)[12] and so on. Graph-structured algorithms by them represented better expression ability than tree structured algorithms by standard GP.

GNP and GAUGE are used for dynamic problems like the control of agents. GAUGE selects pass and output by using the value of memory. On the other hand, GNP selects different output in the same situation by using the process of transition of nodes as a memory. In addition, GNP has been proposed various versions like adding RL[9], applying Automatically Generated Macro Nodes (AGMs)[10] like automatically defined functions (ADFs)[11] and so on.

GRAPE is aimed at constructing arbitrary directed graph structure, handling multiple data types using a data set and converting phenotype of graph structure to genotype of an integer string. It succeeded in constructing some programs for

static problems like sorting[7], the computations of factorial, Fibonacci sequence and exponentiation[5] et al. GRAPE and GNP are expressly similar about some points. First, both methods define the number of available nodes and modify combination of those nodes for searching better combination. Second, a graph structure is able to convert a linear string of integers and evolution progress by modifying it. Moreover, the process of transition of nodes is important to construct more effective algorithms. In this paper, using GRAPE construct algorithms for controlling agents as a dynamic problem because GRAPE can control agents like GNP.

C. Graph structured program evolution

GRAPE constructs graph-structured algorithms automatically by combination of prepared nodes and alteration of nodes. GRAPE is able to reuse nodes and construct the structure of loops by reusing nodes. GRAPE consists of phenotype that is graph structure, genotype that is a linear string of integers and dataset that includes various types of data and is referred by nodes of graph structure. Figure 2 shows an example of the structure of GRAPE.

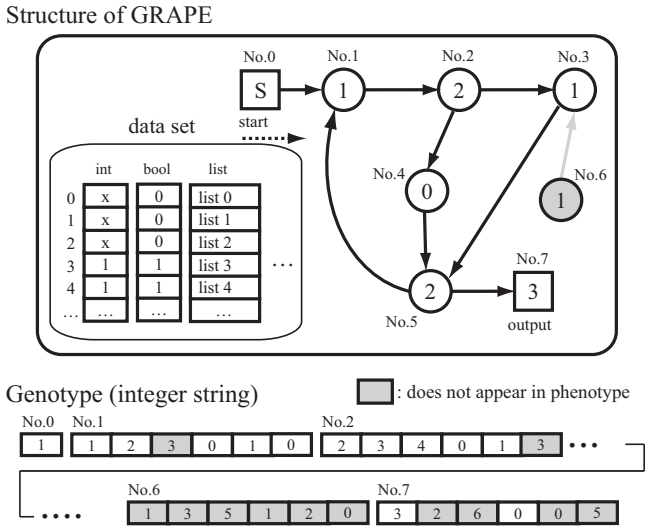


Fig. 2. Structure of GRAPE (phenotype) and the genotype, which denotes a list of node types, connections and arguments. No.6 is the inactive node.

An algorithm that is constructed by GRAPE consists of "Start node", "Processing node", "Branching node" and "Output node". "Start node" is the start of transition of nodes. "Output node" is the end of transition of nodes and outputs data. "Processing node" updates the data for problems by using current that. "Branching node" branches by using that data. Figure 3 shows example nodes in GRAPE. No.1 adds data[0] of an integer data type in dataset to data[1], the sum puts in data[0]. No.2 decides the next node using integer data[0] and data[1]. If data[0] is greater than data[1], connection 1 is chosen, else connection 2 chosen. Each node has information about node types like "Processing" and "Branching node", operation of the node in each node type, arguments that decide using data in dataset at the node. They are data of integers, so

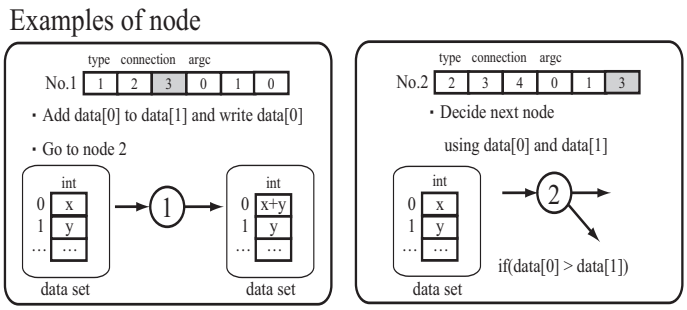


Fig. 3. Examples of processing (left) and branching nodes (right).

genotype of GRAPE is a linear string of integers. Therefore, GRAPE is able to use a typical genetic algorithm (GA)[14]. GRAPE uses uniform crossover and mutation for a string of integers as genetic operators.

It uses the Minimal Generation Gap (MGG) model[13] as a generation alternation model. GRAPE with the MGG model shows a higher performance than it with simple GA[5]. At the MGG model, two individuals remain in parents and children in a generation. One is the elitist individual and the other from roulette-wheel selection. the MGG model can avoid initial convergence and maintain the diversity of the population.

III. Graph structured program evolution for keepaway soccer

We need some modifications of GRAPE for the construction of strategies in keepaway soccer. We allocate the perceptual information to input data and action to "Output node", branching by using input data to "Processing node" and "Branching node". First, the name of objects in this paper shows in Fig4.

- Agent: the keeper in attention.
- K1: the nearest keeper form "Agent".
- K2: the second nearest keeper form "Agent".
- T1: the nearest taker form "Agent".
- T2: the second nearest taker form "Agent".

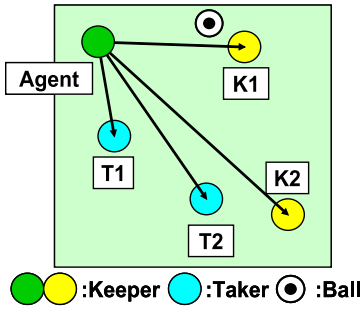


Fig. 4. Name of objects in this paper.

The total number of "Processing node" and "Branching node" is nine. "Processing node" and "Branching node" shows in Table I and Fig. 5. Those nodes branch by the operation of each node, so all prepared nodes of "Branching node" and "Processing node" are only "Branching node". "Branching

TABLE I
"PROCESSING NODE" AND "BRANCHING NODE" FOR KEEPAWAY.

name	operation	number of connection
Keeper1 dist (K1D)	branch by distance from "Agent" to "K1".	4
Keeper2 dist (K2D)	branch by distance from "Agent" to "K2".	4
Taker1 dist (T1D)	branch by distance from "Agent" to "T1".	4
Taker2 dist (T2D)	branch by distance from "Agent" to "T2".	4
Keeper1 Taker dist (K1TD)	branch by distance from "K1" to nearest taker from "K1".	2
Keeper2 Taker dist (K2TD)	branch by distance from "K1" to nearest taker from "K2".	2
Keeper1 angle (K1A)	branch by angle consisting of "K1","Agent"and nearest taker from "K1".	2
Keeper2 angle (K2A)	branch by angle consisting of "K1","Agent"and nearest taker from "K1".	2
Random (Rnd)	branch by probability.	2

TABLE II
"OUTPUT NODE" FOR KEEPAWAY.

name	the action of output	assumption for output
Go To Ball (GB)	intercept the ball.	"Agent" is nearest from the ball.
Hold Ball (HB)	remain stationary with the ball.	"Agent" is able to kick the ball.
Near Pass (NP)	pass to K1.	"Agent" is able to kick the ball.
Far Pass (FP)	pass to K2.	"Agent" is able to kick the ball.
Angle Pass (AP)	pass by using angle information(Fig. 6).	"Agent" is able to kick the ball.
Get Open (GO)	go to free area(Fig. 6).	"Agent" can't output "Go To Ball".
Recieve Move (RM)	get courses of pass(Fig. 6).	"Agent" can't output "Go To Ball".
Last Action (LA)	operate previous "Output node".	depend on previous "Output node"

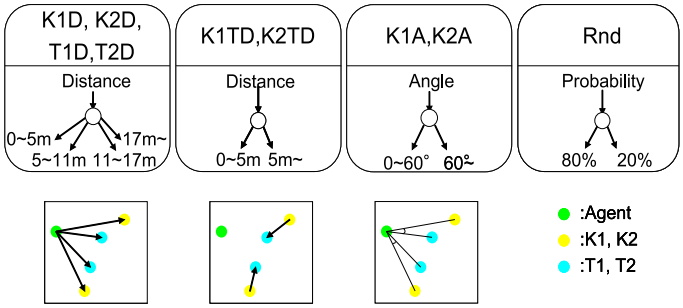


Fig. 5. The detail of "Branching node" and "Processing node".

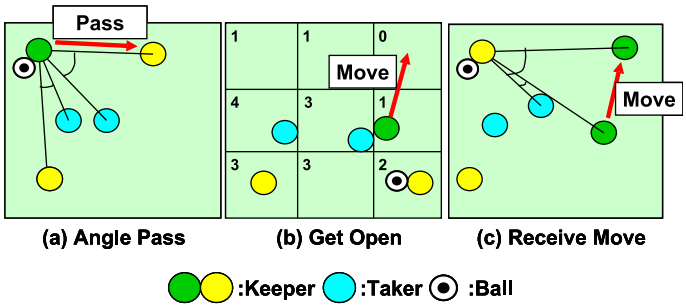


Fig. 6. Outline of some "Output node".

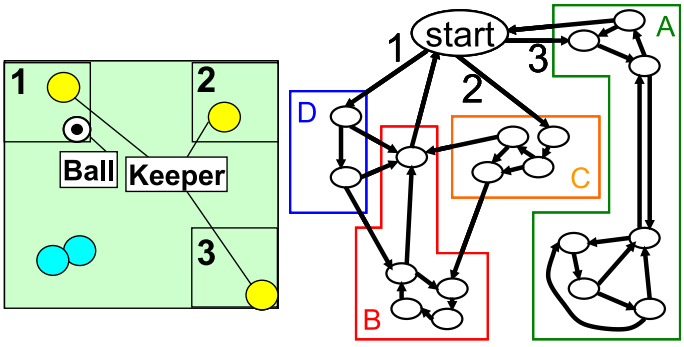
node" uses the perceptual information and random numbers to decide to select connections of node.

"Output node" has assumptions for output. If this is not fulfilled, the transition of nodes continues till this is fulfilled. To set assumptions restrains ineffective actions. It is important for keepaway soccer because an ineffective action causes failure like failing in holding the ball and so on. "Output node" shows in Table II. The number of "Output node" is eight such as "Hold Ball"(Remain stationary with the ball), "Go To Ball"(Intercept a ball) and so on. "Near Pass", "Far Pass", "Hold Ball" and "Go To Ball" output the simple action for the intended target. On the other hand, "Angle Pass", "Get Open" and "Receive Move" go through some procedures before output the action. "Angle Pass" decides the target for pass by the judgement of angle that is formed by "Agent", "K1" or "K2", and the nearest taker from them before pass. "Get Open" evaluates areas that are parts of playing area devided into nine at first. Then, "Agent" goes to the area that got the minimum value of evaluation. The way to evaluation of areas is to add points to each area and to reduce points from each area. Points that are equal to the double number of agents that are keepers except "Agent" and takers are added to the area including those agents. Points that are equal to the number of those agents are added to areas adjoining the area including those agents. In addition, One point is subtracted from the area including "Agent". "Receive Move" gets the segment from "Agent" to keeper having the ball. "Agent" moves to the direction that is vertical against that segment for

increasing the angle that consists of "Agent", a keeper holding the ball and the nearest taker from "Agent". Combination of those prepared nodes constructs graph-structured algorithms. "Output node" shows in Fig. 6.

GRAPE is a method for static problem but keepaway soccer is dynamic problem. Therefore, algorithms by GRAPE output repeatedly in keepaway soccer. Previous "output node" is set up as the start of transition of nodes because that allow agent to decide next output considering previous output and branch on condition. Without miss, keepaway soccer continues proceeding. It is effective for keepaway soccer that an algorithm without the end of transition of nodes continues the transition of nodes.

In addition, we prepare multiple connections of "start node" for keepers and each connection corresponds to each area for keepers starting keepaway soccer in Fig. 7 It allows keepers to be pseudo heterogeneous keepers. A keeper starting from an area of those areas is able to use only one of connections. In Fig. 7(b), Keeper using "1" of connections has the graph structure of "B" and "D". Keeper using "2" of connections has the graph structure of "B" and "C", so two keepers share the structure of "B". However, two keepers do not share all nodes, so they have different algorithms. On the other hand, Keeper using "3" of connections has the structure of "A", so the keeper has a specific algorithm. Keepers use each graph-structured algorithm because agents start different areas. When keepaway soccer starts, a keeper has the clear task that depends on the area of start. Therefore, the pseudo heterogeneous graph-structured algorithm can work effectively for this task.



(a) Numbering of starting points. (b) Numbering of connections of "Start node".

Fig. 7. Example of the pseudo heterogeneous graph structured algorithm.

IV. Experiments

We constructed the strategy of keepers by GRAPE. Proposed method made two kinds of the strategies, one was for homogeneous keepers, and the other was for pseudo heterogeneous keepers. Our experiments matched 3 keepers against 2 takers in a 20[m]×20[m] region. Keepers used strategies constructed by GRAPE. Takers used a hand-coded strategy. Two action patterns of takers were prepared. The nearest taker from the ball intercepts the ball and the other cut in on the course for pass. The strategies were evaluated by the ten times average of time of keeping the ball by keepers. The processing steps of the experiment were as follows:

Step.1 Generate individuals of the first generation and evaluate them.

Step.2 Select two individuals as parents and generate children by crossover and mutation.

Step.3 Evaluate children and select two individuals in parents and children.

Step.4 Replace parents by selected individuals in parents and children.

Step.5 Repeat from Step.2 to Step.4 until the end of generations.

The strategy was constructed according to this process. The parameters of GRAPE were shown in tableIII.

TABLE III
Parameter of GRAPE

Generations	15000
Individuals	20
The model of alternation of generations	MGG
Offspring	10
Genetic operation	Only mutation
Mutation rate	0.05
Max of nodes	35
Max of number of times for transition of nodes	1000

For experiments, we used the soccer server of Robocup that is version 13.2.2[18] and Agent2d of version 2.1.0[17] for making keepers and takers. Agent2d is the template for developing the soccer team for Robocup simulated soccer.

V. Result and Discussion

Figure 8 shows transitions of the keeping time. They are increasing as the number of generations increase. Figure 9 shows the best structure constructed by GRAPE for homogeneous keepers at the last generation.

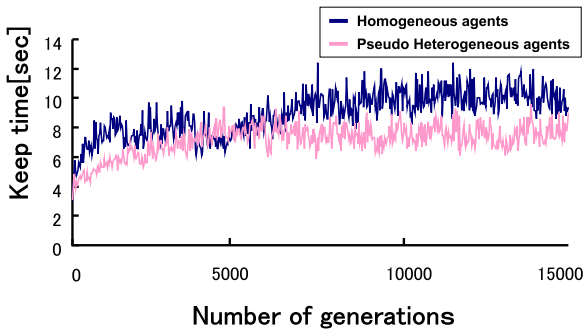


Fig. 8. Transition of time of keeping the ball.

In Fig. 9, "A" contains the main loop that consists of basic output in keepaway soccer. Basic output is continually used some "output node" that are "Hold Ball", "Get Open" and "Go

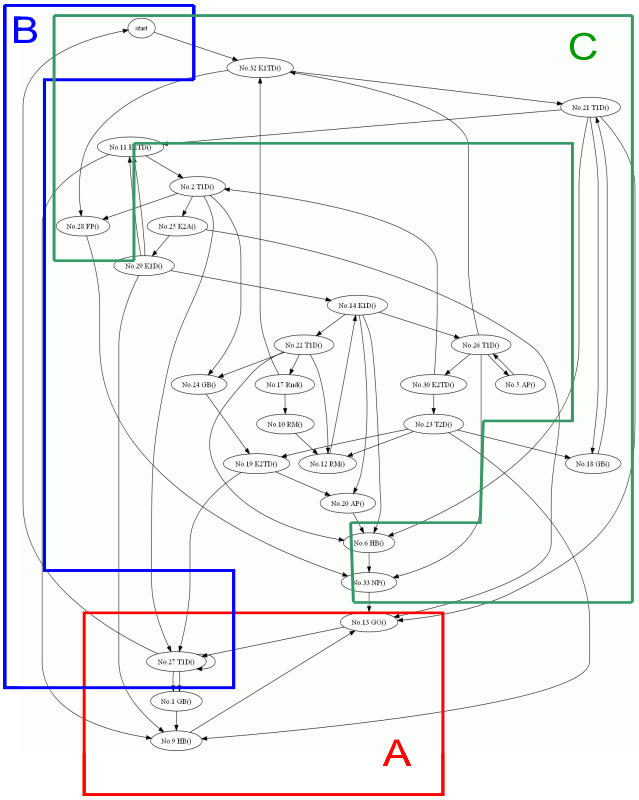


Fig. 9. Graph-structured algorithm for homogeneous agents.

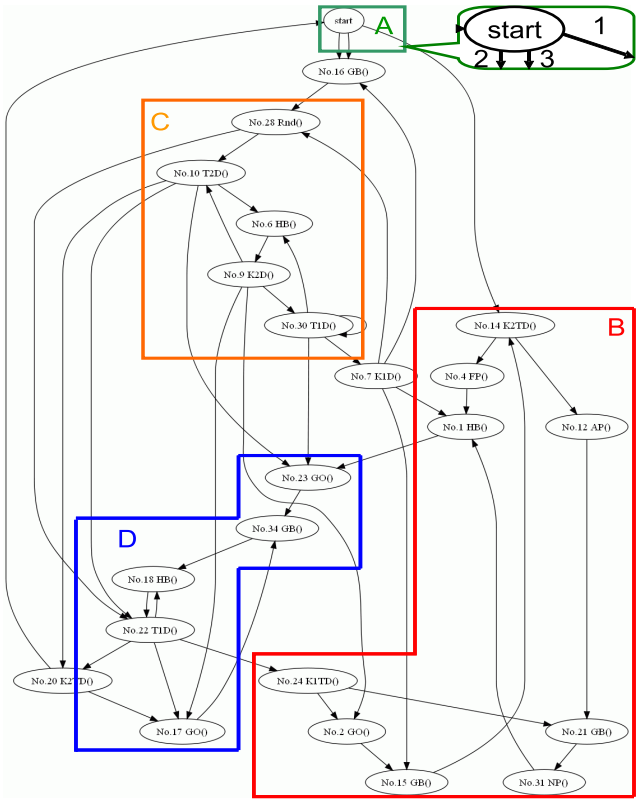


Fig. 10. Graph-structured algorithm for pseudo heterogeneous agents.

To Ball". Then, the transition of "B" exits the main loop of "A" and connects to structure of "C". Structure of "C" works as searching a free keeper and passing the keeper the ball. Keepers get the strategy about attracting takers and passing a free keeper the ball and getting free by Get Open.

In Fig. 10, "1" of connections of "Start node" for the nearest keeper from the ball at the start of keepaway soccer uses structures of "B" and "D". "2" and "3" of connections of "Start node" for keepers receiving the ball at the start of keepaway soccer use structures of "B", "C" and "D". Structures that are used by the transition from "2" and "3" of connections contain structures that are used by the transition from "1" of connections. However, the difference of connecting to the structure come to the difference of main available nodes. "B" in Fig. 10 is the structure for pass. It is connected by "A" and "D" in Fig. 10. If the node transfers from "A" to "B", two kinds of information, distance and angle, use for pass. On the other hand, if the node transfers from "D" to "B", only a kind of information, distance, use for pass. In addition, a nearer keeper from "Agent" is mainly selected as the target for pass. Therefore, the number of failure at the start of keepaway soccer reduces but failure occurs easily as keepaway soccer proceed. In the observation of actions of keepers, the pseudo heterogeneous keepers choose the good target for a pass at the start of keepaway soccer but at proceeding keepaway soccer, choose the bad target for a pass. The keeping time of the pseudo heterogeneous keepers is shorter than that of the homogeneous keepers. Constructing pseudo heterogeneous strategies, GRAPE destructs individual strategies for keepers easily because keepers share nodes but do not use all same nodes. Therefore, the strategy dose not stabilize and the keeping time of pseudo heterogeneous agents is shorter.

VI. Conclusion

In this paper, we constructed the strategy of keepers by GRAPE. We verified that the constructed strategy is effective to control of multiple agents from experiment. Some parts of the graph-structured algorithm acquired the particular role and these are connected effectively. In future works, we aim to get the strategy of achieving more difficult tasks instead of keepaway soccer. In keepaway soccer, we have some future work. In the observation of keepers' actions, keeper often takes no action when pass courses are completely blocked. The reason is the behavior of "Get Open". The behavior of "Get Open" includes the tendency of staying at the same position, so the keeping time increases by improving that tendency. In this paper, we evaluate strategies by only using the keeping time. We need the way of evaluation innovating the consideration of nodes related to misses.

References

[1] Luke, S. and Hohn, C. and Farris, J. and Jackson, G. and Hendler, J., Co-evolving Soccer Softbot Team Coordination with Genetic Programming, Lecture notes in computer science, pp.398-411, 1998.

[2] Hsu, W. H. and Gustafson, S. M., Genetic programming and multi-agent layered learning by reinforcements, Genetic and Evolutionary Computation Conference, New York, NY, 2002.
[3] Stone, P. and Sutton, R. S., Reinforcement Learning for RoboCup Soccer Keepaway, Adaptive Behavior, vol.13, num.3, pp.165-188, 2005.
[4] Stone, P. and Kuhlmann, G. and Taylor, M.E. and Liu, Y., Keepaway soccer: From machine learning testbed to benchmark, Lecture Notes in Computer Science, vol.4020, pp.93, 2006.
[5] Shinichi Shirakawa, Shintaro Ogino, and Tomoharu Nagao: Graph Structured Program Evolution, Proceedings of the Genetic and Evolutionary Computation Conference 2007 (GECCO '07), Vol.2, pp.1686-1693, London, England, 2007.
[6] Shinichi Shirakawa and Tomoharu Nagao: Graph Structured Program Evolution: Evolution of Loop Structures, In Rick L. Riolo and Una-May O'Reilly and Trent McConaghy editors, Genetic Programming Theory and Practice VII, chapter 11, pp.177-194, Springer, 2009.
[7] Shinichi Shirakawa and Tomoharu Nagao: Evolution of sorting algorithm using graph structured program evolution, Proceeding of the 2007 IEEE International Conference on System, Man and Cybernetics (SMC 2007), pp.1256-1261, 2007.
[8] Katagiri, H. and Hirasawa, K. and Hu, J. and Murata, J. and Kosaka, M., Network Structure Oriented Evolutionary Model: Genetic Network Programming-Its Comparison with Genetic Programming, TRANSACTIONS-SOCIETY OF INSTRUMENT AND CONTROL ENGINEERS, vol.38, num.5, pp.485-494, 2002.
[9] Shingo Mabu, Kotaro Hirasawa, and Jinglu Hiu, A Graph-Based Evolutionary Algorithm: Genetic Network Programming (GNP) and Its Extension Using Reinforcement Learning. Evolutionary Computation, Vol.15, No.3, pp.369-398, 2007.
[10] H. Nakagoe, K. Hirasawa, and J. Hu. Genetic network programming with automatically generated variable size macro nodes. In Proceedings of the 2004 IEEE Congress on Evolutionary Computation (CEC 2004), pp.713-719, Portland, Oregon, 20-23 June 2004. IEEE Press.
[11] J. R. Koza, Genetic Programming II: Automatic Discovery of Reusable Programs. MIT Press, Cambridge, MA, USA 1994.
[12] Hiroaki Kataoka, Akira Hara and Tomoharu Nagao: Genetic Automata Generation; GAUGE, Information Processing Society of Japan, Vol.44, No.12, pp.3232-3241, 2003. [In Japanese]
[13] H. Satoh, M. Yamamura, and S. Kobayashi. Minimal generation gap model for considering both exploration and exploitations In Proceedings of the IIZUKA '96, pp.494-467, 1996.
[14] John H. Holland, Adaptation in Artificial and Natural Systems, The University of Michigan Press, 1975.
[15] John R. Koza, Genetic programming: On the programming of computers by means of natural selection, MIT Press, 1992.
[16] Sutton R. S. and Barto A. G., Reinforcement learning, MIT Press, 1998.
[17] Robocup tools, http://sourceforge.jp/projects/rctools.
[18] Soccer Server, http://sourceforge.jp/projects/sserver.