Evolving Directed Graphs with Artificial Bee Colony Algorithm

Xianneng Li
Graduate School of Information, Production, and Systems
Waseda University
Kitakyushu, Japan
Email: sennou@asagi.waseda.jp

Guangfei Yang
Institute of System Engineering
Dalian University of Technology
Dalian, China
Email: gfyang@dlut.edu.cn

Kotaro Hirasawa
Graduate School of Information, Production, and Systems
Waseda University
Kitakyushu, Japan
Email: hirasawa@waseda.jp

***Abstract*—Artificial bee colony (ABC) algorithm is a relatively new optimization technique that simulates the intelligent foraging behavior of honey bee swarms. It has been applied to several optimization domains to show its efficient evolution ability. In this paper, ABC algorithm is applied for the first time to evolve a directed graph chromosome structure, which derived from a recent graph-based evolutionary algorithm called genetic network programming (GNP). Consequently, it is explored to new application domains which can be efficiently modeled by the directed graph of GNP. In this work, a problem of controlling the agents's behavior under a well-known benchmark testbed called Tileworld are solved using the ABC-based evolution strategy. Its performance is compared with several very well-known methods for evolving computer programs, including standard GNP with crossover/mutation, genetic programming (GP) and reinforcement learning (RL).**

***Keywords*-artificial bee colony; genetic network programming; directed graph; agent control; computer programs**

I. Introduction

Swarm intelligence is an evolutionary computation (EC) technique that studies the collective behaviors of decentralized, natural, and artificial systems. Numerous algorithms have been developed in this area which have been shown to outperform other algorithms in many applications, such as particle swarm optimization (PSO) [1], ant colony optimization (ACO) [2] and artificial immune systems (AIS) [3], etc.

Artificial bee colony (ABC) [4] algorithm is a relatively new swarm-based technique that simulates the foraging behavior of honey bee swarm. ABC algorithm imitates three types of bees to search the space, i.e., employed bees, onlooker bees and scout bees. The employed bees are associated with the existing food sources and are designed to modify the associated food sources to explore the search space. Employed bees share their nectar information with the onlooker bees, where the onlooker bees select some of the food sources based on their qualities, which are further modified to explore the search space. If some food sources are recognized to be obsolete, they are abandoned and the scout bees are sent to search (generate) a new food source. ABC algorithm is fairly simple and easy for implementation.

Numerous studies have been conducted to compare the performance of ABC algorithm with the other EC techniques, such as genetic algorithm (GA), PSO, ACO and differential evolution (DE) [5] on the problems of function optimization, which demonstrated its competitive performance with an advantage of requiring fewer control parameters [6], [7].

Like the other swarm-based algorithms, ABC algorithm is directly applied to optimize the variables of the given problems. In other words, there is no chromosome encoding scheme in ABC algorithm, unlike the evolutionary algorithms (EAs) such as GA [8] with bit-string chromosomes, and genetic programming (GP) [9] with tree chromosomes, etc. This results in that up to date, ABC algorithm is generally applied to solve the problems of function optimizations or some related variants.

Recently, researchers have explored ABC algorithm to evolve tree structure based chromosomes, like that of GP. This extension, called ABC programming (ABCP) [10], explicitly employs the advantages of ABC algorithm to evolve the tree based candidate solutions of GP, which is capable of exploring to solve more different sorts of problems. It is shown that ABCP can achieve superior performance on symbolic regression problems comparing with the state-of-the-art GP algorithms using traditional crossover and mutation.

In EAs, there are recently some interests in developing more complicated chromosome encoding schemes to extend GA and GP, like graph structure chromosomes [11], [12]. Explicitly, the more complicated the chromosomes are, the higher the expression ability of modeling complicated systems is. Genetic network programming (GNP) [13], [14], [15], originated by K. Hirasawa, is such a kind of graph-based EA that develops a directed graph based chromosome encoding scheme to model complicated systems. GNP is a fairly simple algorithm that directly applies the traditional crossover and mutation to evolve the directed graph for global search. It has been successfully utilized to solve many different kinds of complicated problems, such as the problems of controlling the agents' behavior [16], robot control [17], [18], [19], data mining [20], [21], financial

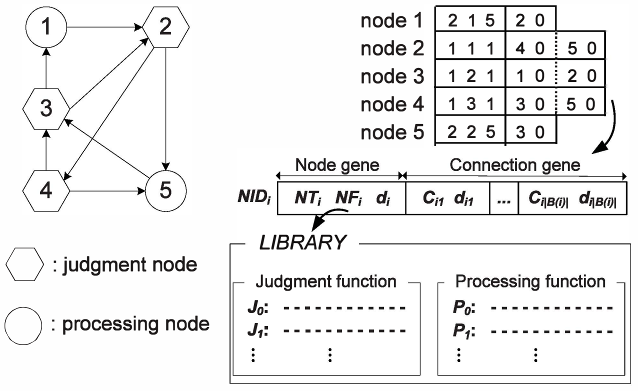

Figure 1: Directed graph based chromosome encoding

engineering [22], [23], and so on.

In this paper, ABC algorithm is introduced to GNP that replaces its traditional crossover/mutation based evolution mechanisms by the foraging behavior of honey bees. With the unique directed graph based chromosome encoding, ABC algorithm is explored to solve complicated problems, that is, evolving the strategies on the problems of controlling the agents' behavior. The performance of the proposed algorithm is compared with several very well-known methods evolving computer programs, including standard GNP with crossover/mutation, genetic programming (GP) and reinforcement learning (RL), under a well-known benchmark testbed called Tileworld [24].

II. Directed Graph Based Chromosome Encoding

The most distinguished feature of GNP is that it develops a directed graph structure to encode the chromosomes, as shown in Figure 1. The directed graph consists of judgment and processing nodes. The nodes of GNP can be connected arbitrarily, while these node connections are subject to evolution, which allows the flexible expression and high evolution ability for modeling some complex problems.

Judgment nodes work as "IF-THEN" decision-making functions to judge the environments. Processing nodes without any conditional branch consist of the actions. By separating judgment and processing, the evolution can efficiently evolve the compact programs by only transiting the necessary judgments and processing. Such a property named transition by necessity [16] makes GNP an efficient rule generator for decision-making.

A. Chromosome encoding

Though illustrated as a complicated graph, the directed graph is encoded into bit-strings, defined by a tuple $G = (N_{\text{node}}, B, \mathit{LIBRARY})$, where

- N_{node}: the set of nodes in one graph
- B: the set of branches in one graph

```
1  initialization();
2  for individual $i \leftarrow 1$ to $popSize$ do
3      execute($i$);
4  end
5  for generation $g \leftarrow 1$ to $maxGeneration$ do
6      for employBee $i \leftarrow 1$ to $popSize$ do
7          $i'$ =sendEmployedBee($i$);
8          execute($i'$);
9          greedySelection();
10     end
11     for onlookerBee $t \leftarrow 1$ to $popSize$ do
12         $i$ =tournamentSelection();
13         $i'$ =sendOnlookerBee($i$);
14         execute($i'$);
15         greedySelection();
16     end
17     sendScoutBee();
18 end
```

Algorithm 1: Basic procedure of GNP-ABC

- *LIBRARY*: the set of judgment/processing functions

Each node, i.e., node i, is defined by a tuple $(NT_i, NF_i, d_i, B(i), C_i)$.

- NT_i: the node type, where 1 or 2 for judgment or processing node, respectively.
- NF_i: the function of node i loaded from the *LIBRARY*.
- d_i: the time delay spent on the judgment or processing of node i.
- $B(i)$: the set of branches of node i.
- C_i: consisting of a set of C_{ik} and d_{ik}. C_{ik} indicates the node connected from node i by its k_{th} branch, and d_{ik} is the time delay spent on this node transition ($1 \le k \le |B(i)|$).

When executing a GNP individual, we start the node transition by a predefined start node (generally a judgment node), where the hereafter node transitions are carried out based on the interaction with the environment.

B. Search variables

The aim of evolution is to find the optimal directed graph by evolving the node connections C of each node, where the other parameters are defined in advance and fixed. In other words, the number of search variables corresponds to the number of branches in one graph, that is $|B|$.

For example, there are total 8 search variables (branches) in the directed graph of Figure 1. Evolution pressure is employed to evolve the directed graph by changing the node connections. Each node's branch is possible to connect to any other node of the graph except itself.

III. Artificial Bee Colony Based Evolution

In standard GNP, the node connections are determined by traditional crossover and mutation. In this paper, we develop a new evolution mechanism based on ABC algorithm to replace the traditional crossover and mutation. The detailed pseudo-code of the proposed algorithm, named GNP with ABC (GNP-ABC) is given in Algorithm 1.

A. Food source (individual)

Initially, GNP-ABC generates a population of food sources, where each food source is an individual. The initial $popSize$ individuals are randomly generated by randomly determining the node connections of the directed graph.

B. Employed bee phase

At the beginning of each generation, $popSize$ employed bees are sent to the food sources. Each employed bee i is associated with a particular food source i (individual). It searches a new individual i' in the neighboring space of its existing individual.

In order to create the new individual i', the employed bee applies an information sharing mechanism. Firstly, it randomly selects an individual called $neighbor$. Afterwards, one node j in the directed graph is randomly selected. All the node connections of j, i.e., C_{jk} ($k = 1, 2, ..., |B(j)|$), in individual i is replaced by that of individual $neighbor$ to create the new individual i'. (This process corresponds to line 7 of Algorithm 1)

After the creation of the new individual i', it is executed to evaluate the fitness. A greedy selection is hereafter applied. If the quality of i' is better than i, the new individual i' is survived in the next generation by replacing i. Otherwise, the old individual i is remained.

C. Onlooker bee phase

After sending the employed bees to search the neighboring space, onlooker bees are sent to further explore the promising region of the search space.

Firstly, each onlooker bee applies tournament selection to select one individual i to process. A new individual i' is created by modifying the selected individual i as follows.

```
1) Select node $j \leftarrow 1$;
2) Produce a random value $rand \in [0, 1]$;
3) If $rand < modif\_rate$
       Randomly set $C_{jk}$, ($k = 1, 2, ..., |B(j)|$);
4) Set $j \leftarrow j + 1$;
5) Go back to 2) until $j > |N_{\text{node}}|$.
```

Here, $modif\_rate$ is a user-defined parameter to control the modification rate, which should be generally small.

Similar to the employed bees, onlooker bees apply greedy selection to select the better individual between i and i' to the next generation.

D. Scout bee phase

If the selected individual i is not improved by the employed bees and onlooker bees for a certain number of trials, i.e., defined by a threshold $limit$, a scout bee is sent to abandon this individual and generate a new individual.

The new individual can be generated in many ways. One is by random, which is used in the original ABC algorithm. However, in our study we find that it is not efficient when evolving the directed graph of GNP. In this paper, we replace the abandoned individual by another existing individual in the population using tournament selection. In order to improve the population diversity, the selected individual is slightly modified using the process shown in the onlooker bee phase.

IV. Experimental Study

The proposed algorithm is applied to solve a new problem different from the existing ABC algorithms, that is the problem of controlling the agents' behavior. A well-known benchmark testbed called Tileworld [24] is selected to conduct the experimental study.

Tileworld is a parameterized testbed, which consists of a grid of cells including agents, tiles and holes. It has been widely used for the development of multi-agent systems and the evolution of computer programs (control strategies) [25], [26], [27], [28], [29]. A $12 * 12$ grid world consisting of 3 tiles and holes is designed for experiments, as shown in Figure 2. In the world, 3 agents are created, which are capable of judging the surrounding environment and interacting with it by taking the appropriate actions. The target of Tileworld is to control the agents to cooperate with each other to push the tiles into the holes as many as possible, using steps as less as possible, or pushing the tiles towards the holes as close as possible if there are remaining tiles that cannot be pushed into the holes in a given steps. Accordingly, the fitness function of Tileworld can be defined as follows.

$$f = 100DT + 3(ST - S_{\text{used}}) + 20\left[\sum_{t \in Tile}\Big(D(t) - d(t)\Big)\right], \tag{1}$$

where,
DT: number of tiles that have been pushed into the holes.
ST: user-defined constrained steps.
S_{used}: number of steps that have been used.
$Tile$: set of tiles.
$D(t)$: original distance from tile t to its nearest hole.
$d(t)$: distance from tile t to its nearest hole after ST steps.

The user-defined constrained steps ST can be set flexibly to define the problem complexity of Tileworld. In this paper, ST is set to 60.

The agent is designed to have 8 sensor abilities to perceive the state of its neighboring cells and the direction information of the tiles and holes, as shown in Table I. Based on the sensor results, the agent can move forward, turn left, turn right or stay (4 processing functions).

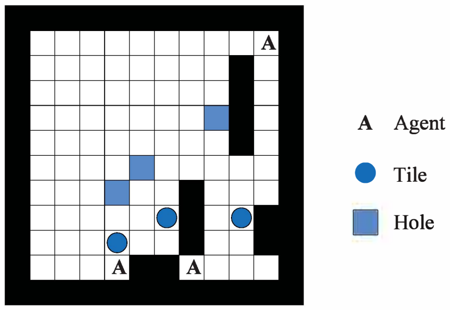

Figure 2: Tileworld

Table I: Judgment functions for Tileworld

Function	Description	Content of Branches
J_1	`Judge Forward`	0: Empty 1: Obstacle
J_2	`Judge Backward`	2: Tile 3: Hole
J_3	`Judge Left`	
J_4	`Judge Right`	4: Agent
J_5	Judge the `Direction` of the nearest `Tile` from the agent	0: Forward 1: Backward
J_6	Judge the `Direction` of the nearest `Hole` from the agent	2: Left 3: Right
J_7	Judge the `Direction` of the nearest `Hole` from the nearest `Tile`	4: Cannot find
J_8	Judge the `Direction` of the `Second` nearest `Tile` from the agent	

A. *Compared algorithms and parameter settings*

Several very well-known methods for evolving computer programs, including standard GNP with crossover/mutation [13], genetic programming (GP) [9] and reinforcement learning (RL) [30], are selected for comparison to evaluate the performance of the proposed GNP-ABC algorithm.

1) GNP and GNP-ABC: GNP and GNP-ABC uses a directed graph to design the chromosomes. The directed graph is defined based on the suggestions of literature [15]. The node size $|N_{\text{node}}| = 60$, including 40 judgment nodes and 20 processing nodes (5 nodes per each judgment and processing function). GNP applies uniform crossover and mutation [13] to evolve the directed graphs, which include two evolutionary parameters: crossover rate $p_c = 0.1$ and mutation rate $p_m = 0.01$. In the proposed GNP-ABC, ABC algorithm is designed to replace crossover and mutation. The parameters of GNP-ABC include the modification rate $modif\_rate = 0.02$ in onlooker bee phase and the threshold $limit = 200$ in scout bee phase. Tournament selection with size 2 is used for both GNP and GNP-ABC.

2) GP: GP uses tree structure for chromosome encoding. The maximum tree depth of GP is defined to 4. FULL method is used to initialize the GP individuals. The crossover and mutation rates of GP is set to $p_c = 0.9$ and $p_m = 0.01$.

3) RL: Sarsa learning (Sarsa) [30] is selected as a state-of-the-art RL algorithm. In Sarsa, the learning rate is set to 0.2, the discount factor is set to 0.9 and $\varepsilon-$greedy with $\varepsilon = 0.1$ is used for action selection.

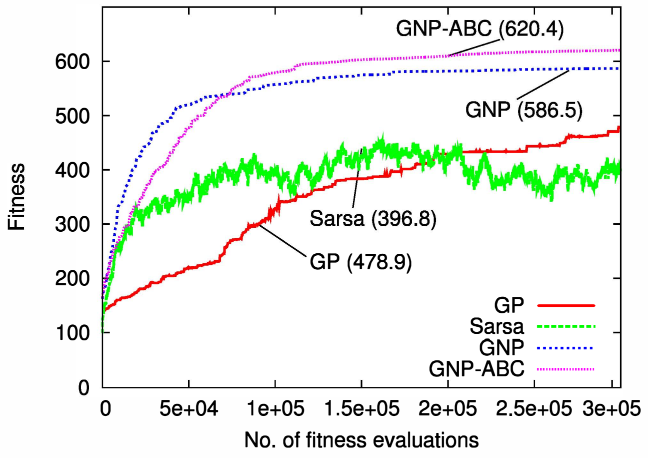

Figure 3: Fitness curves

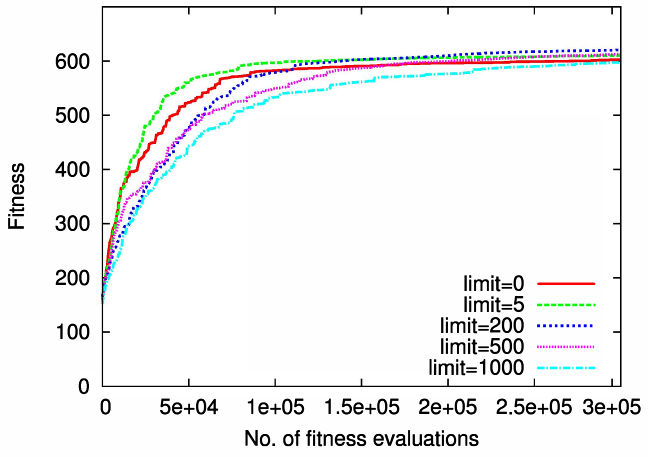

Figure 4: Fitness curves under different $limit$ values

4) Additional parameters: The population sizes $popSize$ of GNP, GP and GNP-ABC are set to 300. The terminal condition of each algorithm is defined by the maximum number of fitness evaluations, where 300 000 is used in this paper. All the results shown in this paper is the average values of 30 independent experiments to eliminate the evolutionary randomness.

Table II: Fitness values and student t-test results (the † symbol in the t-test column indicates that there is statistically significant difference between GNP-ABC and the compared algorithm)

	Fitness value	Standard deviation	t-test
GP	478.9	151.6	†
Sarsa	396.8	114.3	†
GNP	586.5	69.8	†
GNP-ABC	**620.4**	**8.4**	

B. Experimental results

In the experimental environment of Figure 2, theoretically the agents need at least 15 steps to push all tiles into holes. Accordingly, the maximum fitness value is 635.

The fitness curves of the compared algorithms are plotted in Figure 3. It is realized that the directed graph structure is capable of modeling the complicated strategies for Tileworld efficiently, where GNP and GNP-ABC achieve much better fitness values than that of GP and Sarsa.

Comparing with standard GNP with crossover and mutation, it is clear that the proposed GNP-ABC achieves better performance. This indicates that ABC algorithm based evolution can achieve higher evolution efficiency than the traditional crossover and mutation based evolution when evolving the directed graph of GNP. When looking at the landscapes of the fitness curves, it is found that GNP-ABC shows slower evolution speed than GNP in the early generations. This is due to that in GNP-ABC, both of the employed bees and onlooker bees applies the greedy selection to produce new individuals, which somehow restrict the exploration ability of evolution. Scout bees start to affect the evolution by exploring the search space only after a certain number of unsuccessful trials, i.e., $limit = 200$. However, with the deepening of the evolution, GNP-ABC can gradually find better results than GNP.

C. Statistical analysis

The detailed fitness values and the standard deviations are listed in Table II. The results show that the proposed GNP-ABC can achieve the highest fitness value with smallest standard deviations, which indicates its powerful and stable evolution ability.

Student t-test (two-tailed, paired) is applied to evaluate the statistical significance of the experimental results. In t-test, 95% confidence level is used, where the test results show that GNP-ABC outperforms the compared algorithms with statistical significance.

D. Impact of scout bees

We further analyze the impact of the scout bees on the performance of GNP-ABC.

In GNP-ABC, if one individual is not improved after a certain number of trials by sending employed bees and onlooker bees, this individual is considered as an abandoned individual. The abandoned individual will be eliminated from the population, and a scout bee will be sent to generate a new individual. In other words, the scout bee phase works as a exploration strategy to try to explore the search space.

The signal of sending scout bees is designed by a parameter $limit$, which defines the number of unsuccessful trials by employed bees and onlooker bees. Explicitly, if $limit$ is set towards the maximum number of generations, the scout bees will be less sent. On the other hand, if $limit$ is set towards 0, the scout bees will be more frequently sent.

Figure 4 plots the fitness curves of GNP-ABC under different $limit$ values. In the studied experiments, the maximum number of generation is 1000. Therefore, if $limit = 1000$, the scout bee phase will never be activated. In that sense, the exploration is not encouraged in GNP-ABC. The fitness curves show that GNP-ABC with $limit = 1000$ achieves the worst results, and its evolution speed is relatively slow in the early generations.

With the decrease of $limit$ values, the evolution speed will be increased in the early generations, since more exploration is encouraged, which can significantly improve the population quality from its initial stage. For example, when $limit = 0$, scout bees will be sent regardless of whether the individuals are improved or not. This will largely increase the population diversity, however, which will also cause too much evolutionary randomness that make GNP-ABC hard find the optimal solutions.

Under different settings, it is found that $limit = 200$ can achieve the best balance of exploitation-exploration in GNP-ABC that realizes the best performance.

V. Conclusions and Future Work

In this paper, artificial bee colony (ABC) algorithm is applied to evolve a directed graph chromosome structure developed in genetic network programming (GNP). The complicated directed graph is capable of modeling many complicated systems, such as the problems of controlling the agents' behavior studied in this paper. With the ABC based evolution, the proposed algorithm, called GNP-ABC, can significantly improve the evolution efficiency of standard GNP which applies the traditional crossover and mutation for evolution. In the future, the proposed GNP-ABC will be improved by further developing the suitable ABC variants to balance the exploitation-exploration of evolution, as well as being applied to more complicated problems.

References

[1] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 1, pp. 58–73, 2002.

[2] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *Evolutionary Computation, IEEE Transactions on*, vol. 1, no. 1, pp. 53–66, 1997.

[3] L. N. De Castro and F. J. Von Zuben, "Learning and optimization using the clonal selection principle," *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 3, pp. 239–251, 2002.

[4] D. Karaboga and B. Basturk, "On the performance of artificial bee colony (abc) algorithm," *Applied soft computing*, vol. 8, no. 1, pp. 687–697, 2008.

[5] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[6] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm," *Journal of global optimization*, vol. 39, no. 3, pp. 459–471, 2007.

[7] D. Karaboga and B. Akay, "A comparative study of artificial bee colony algorithm," *Applied Mathematics and Computation*, vol. 214, no. 1, pp. 108–132, 2009.

[8] J. H. Holand, *Andaptation in Natural and Artificial Systems.* Ann-Arbor, University of Michigan Press, 1975.

[9] J. R. Koza, *Genetic Programming, on the Programming of Computers by Means of Natural Selection.* MIT Press, 1992.

[10] D. Karaboga, C. Ozturk, N. Karaboga, and B. Gorkemli, "Artificial bee colony programming for symbolic regression," *Information Sciences*, vol. 209, pp. 1–15, 2012.

[11] A. Teller and M. Veloso, "PADO: Learning tree structured algorithms for orchestration into an object recognition system," Carnegie Mellon University, Tech. Report CMU-CS-95-101, 1995.

[12] J. F. Miller and P. Thomson, "Cartesian genetic programming," in *Proc. of the Eur. Conf. on Genetic Programming*, 2000, pp. 121–132.

[13] K. Hirasawa, M. Okubo, H. Katagiri, J. Hu, and J. Murata, "Comparison between genetic network programming (GNP) and genetic programming (GP)," in *Proc. of the IEEE Congres. Evol. Comput.*, 2001, pp. 1276–1282.

[14] S. Mabu, K. Hirasawa, and J. Hu, "A graph-based evolutionary algorithm: Genetic network programming (GNP) and its extension using reinforcement learning," *Evol. Comput.*, vol. 15, no. 3, pp. 369–398, 2007.

[15] X. Li, S. Mabu, and K. Hirasawa, "A novel graph-based estimation of distribution algorithm and its extension using reinforcement learning," *Evolutionary Computation, IEEE Transactions on*, vol. 18, no. 1, pp. 98–113, 2014.

[16] X. Li, W. He, and K. Hirasawa, "Genetic network programming with simplified genetic operators," in *Neural Information Processing*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, vol. 8227, pp. 51–58.

[17] X. Li, B. Li, S. Mabu, and K. Hirasawa, "A novel estimation of distribution algorithm using graph-based chromosome representation and reinforcement learning," in *Proc. of the IEEE Congress on Evol. Comput.*, 2011, pp. 37–44.

[18] X. Li, S. Mabu, and K. Hirasawa, "Use of infeasible individuals in probabilistic model building genetic network programming," in *Proc. of the Genetic and Evol. Comput. Conf.*, 2011, pp. 601–608.

[19] ——, "An extended probabilistic model building genetic network programming using both of good and bad individuals," *IEEJ Trans. Electrical and Electronic Engineering*, vol. 8, no. 4, pp. 339–347, 2013.

[20] X. Li, S. Mabu, H. Zhou, K. Shimada, and K. Hirasawa, "Genetic network programming with estimation of distribution algorithms and its application to association rule mining for traffic prediction," in *Proc. of the ICCAS-SICE*, 2009, pp. 3457–3462.

[21] ——, "Genetic network programming with estimation of distribution algorithms for class association rule mining in traffic prediction," in *Proc. of the IEEE Congres. Evol. Comput.*, 2010, pp. 2673–2680.

[22] Y. Chen and X. Wang, "A hybrid stock trading system using genetic network programming and mean conditional value-at-risk," *European Journal of Operational Research*, no. 0, pp. –, 2014, (in press).

[23] X. Li, W. He, and K. Hirasawa, "Creating stock trading rules using graph-based estimation of distribution algorithm," in *Proc. of the IEEE Congres. Evol. Comput.*, 2014, pp. 731–738.

[24] M. E. Pollack and M. Ringuette, "Introducing the tile-world: Experimentally evaluating agent architectures," in *Proc. of the Conf. of the American Association for Artificial Intelligence*, 1990, pp. 183–189.

[25] H. Iba, "Multi-agent reinforcement learning with genetic programming," in *Proc. of the Annu. Genetic Programming Conf.*, 1998, pp. 167–175.

[26] T. Eguchi, K. Hirasawa, J. Hu, and N. Ota, "A study of evolutionary multiagent models based on symbiosis," *IEEE Trans. Syst., Man, Cybern. B*, vol. 36, no. 1, pp. 179–193, 2006.

[27] X. Li, S. Mabu, and K. Hirasawa, "Towards the maintenance of population diversity: A hybrid probabilistic model building genetic network programming," *Trans. of the Japanese Society for Evol. Comput.*, vol. 1, no. 1, pp. 89–101, 2010.

[28] X. Li and K. Hirasawa, "A learning classifier system based on genetic network programming," in *Proc. of the IEEE Int'l Conf. on Syst., Man, Cybern.*, 2013, pp. 1323–1328.

[29] ——, "Continuous probabilistic model building genetic network programming using reinforcement learning," *Applied Soft Computing*, 2014, in press.

[30] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction.* MIT Press, 1998.