Probabilistic Model Building Genetic Network Programming Using Multiple Probability Vectors

Xianneng Li *Student Member, IEEE*, Shingo Mabu *Member, IEEE*,
Manoj K. Mainali and Kotaro Hirasawa *Member, IEEE*
Graduate School of Information, Production and Systems, Waseda University, Japan
Email: sennou@asagi.waseda.jp, mabu@aoni.waseda.jp, mkmainali@fuji.waseda.jp and hirasawa@waseda.jp

***Abstract*—As an extension of GA and GP, a new evolutionary algorithm named Genetic Network Programming (GNP) has been proposed. GNP uses the directed graph structure to represent its solutions, which can express the dynamic environment efficiently. The reusable nodes of GNP can construct compact structures, leading to a good performance in complex problems. In addition, a probabilistic model building GNP named GNP with Estimation of Distribution Algorithm (GNP-EDA) has been proposed to improve the evolution efficiency. GNP-EDA outperforms the conventional GNP by constructing a probabilistic model by estimating the probability distribution from the selected elite individuals of the previous generation. In this paper, a probabilistic model building GNP with multiple probability vectors (PMBGNP$_M$) is proposed. In the proposed algorithm, multiple probability vectors are used in order to escape from premature convergence, and genetic operations like crossover and mutation are carried out to the probability vectors to maintain the diversities of the populations. The proposed algorithm is applied to the controller of autonomous robots and its performance is evaluated.**

I. Introduction

A new directed graph based evolutionary algorithm named Genetic Network Programming (GNP) has been proposed recently [1][2][3]. Multiple nodes and their branches are used to construct the directed graph structures to represent the solutions of GNP. The node transitions of GNP can memorize the past judgment and processing information in the network flow implicitly and make quite compact structures.

Many studies on evolutionary computation have been executed, such as Genetic Algorithm (GA) [4], Genetic Programming (GP) [5] and Evolutionary Programming (EP) [6]. Compared with these classical evolutionary algorithms, the directed graph structure based GNP has some properties to handle the problems in dynamic environments efficiently and effectively: (1) The directed graph structure of GNP are composed of a number of judgment and processing nodes, which can improve the expression ability of dynamic environments. (2) A fixed number of nodes and branches is used to build the structures of GNP which can avoid the bloating problem compared with GP. (3) The past history of node transitions affects the current node to be executed, which works like the implicit memory function. And the good node transitions could be carried out by evolution, which works like the Automatically Defined Functions in GP. (4) EP uses finite state machines to define transition rules for all combinations of states and possible inputs. Therefore, when dealing with the complex problems in dynamic environments, the structure of EP becomes quite large and complex. However, the nodes of GNP are connected and transited by necessity. GNP can only select the necessary information for the task to generate partially observable processes. Therefore, the structure of GNP could be relatively small and flexible even when simulating the dynamic environments.

However, as many other evolutionary algorithms, GNP evolves its individuals by genetic operators such as crossover and mutation, which can express and reconstruct the *Building Blocks* explicitly. The major drawback of GNP is that so many trials must be executed to search the global optimum. The evolution performance of GNP is profoundly influenced by the parameters settings of crossover probability and mutation probability. The crossover and mutation recombine and preserve the existing *Building Blocks* stochastically. However, the stochastic theorem of crossover and mutation might decompose the *Building Blocks*, which lead to high computational cost. Therefore, one possible way for overcoming such imperfection is to prevent the breakage of *Building Blocks* reducing the generation of invalid individuals. Therefore, a novel probabilistic model building GNP named GNP with Estimation of Distribution Algorithm (GNP-EDA or PMBGNP) [7][8] has been proposed to improve the ability of generating valid offspring, which combines GNP and EDA [9] to avoid the break down of *Building Blocks* and to improve the evolution ability.

PMBGNP allows us to take into account the dependencies between nodes, and therefore shows more suitability for complex problems which could be decomposed to sub-problems implicitly, where there are a number of *Building Blocks* spreading in searching space. In PMBGNP, a probabilistic model is constructed by estimating the probability distribution from the selected elite individuals of the previous generation. A theoretical foundation in statistical learning is taken for modeling the probability vector. The further exploration of searching space is guided by such probability vector, where the interdependencies between different nodes are studied and explicitly represented, which is different from the conventional crossover and mutation.

PMBGNP inherits the advantages of GNP. Comparing with the most current EDA algorithms designed based on GA or GP, the gene structure of PMBGNP has higher expression ability to solve some complex real-world applications.

On the other hand, most of the current EDA based evolu-

tionary algorithms suffer from the problems that the diversity of the genetic information will be significantly decreased in the generated population when the population size is not large enough. Therefore, the EDA population size is usually set at large when solving the problems [10], which will decrease the efficiency of evolution. Some studies investigate that applying mutation operation could increase the diversity of population with small size. For example, Handa proposed some simple mutation operations to incorporate the UMDA, MIMIC and EBNA algorithms and get better performance in theoretical function optimization problems [11]. The conventional PMBGNP inherits such problem.

This paper proposes PMBGNP using multiple probability vectors (PMBGNP$_M$) as a new algorithm to escape from the local optimum of final solutions. Moreover, genetic operations like crossover and mutation are carried out to multiple probability vectors in order to improve the exploration ability of PMBGNP$_M$. The motivation of this innovative algorithm is that: firstly, it evolves multiple populations of GNP by constructing multiple probability vectors that take into account deeper diversities rather than a single probability vector of the conventional PMBGNP. Secondly, it also applies genetic operations like crossover and mutation to the constructed multiple probability vectors, where the exploration of searching space and diversity of population could be improved. With the characteristics of PMBGNP$_M$, the evolution ability of escaping from the local convergence could be enhanced without setting of large population size. In the simulations, the proposed algorithm is applied to the controller of a kind of autonomous robots, Khepera robot [12]. A comparative study of PMBGNP$_M$, PMBGNP and GNP is carried out.

The rest of this paper is organized as follows. In the next section, the proposed algorithm is described in details. Section III presents the results of the simulations. Section IV concludes this paper.

II. Probabilistic Model Building Genetic Network Programming Using Multiple Probability Vectors (PMBGNP$_M$)

A. Basic structure of PMBGNP$_M$

PMBGNP$_M$ is an extension of GNP, whose structure is expressed by a directed graph which is composed of a number of nodes. Fig. 1 shows the basic structure of PMBGNP$_M$, as well as GNP. One start node, fixed number of judgment nodes and processing nodes compose of the graphical structure. The transition starts from the start node, and each judgment node which works as "*if-then*" type decision making functions judges the information from environments to make a decision to transit to the next node. Processing nodes preserve the action functions to determine the actions. Moreover, judgment nodes have conditional branches connecting to different nodes due to the different judging results, while processing nodes have no conditional branch.

There are time delays on PMBGNP$_M$. Three types of time delays are designed in the graphical structure. One is devoted to the judgment nodes, another is designed for the node transition and the last is for the processing nodes. The motivation of time delays is to simulate the interaction between agents and environments explicitly. For example, when an agent moves in an environment, some time it is needed to be spent on judging the environment, preparing for actions and taking actions. The values of time delays can be predefined. In this paper, the time delay of judgment nodes is set at 1 time unit, that of node transition is set at 0 and that of processing node is set at 5 time units. The robot will take one step of action when 10 or more time units are used. For example, after executing four judgments and one processing, if another one processing is executed, the total time units becomes 14, which lead to end one step. On the other hand, the simulated agents end when the end condition is satisfied, that is the time step exceeds its threshold or the given task finishes by PMBGNP$_M$. Therefore, with the design of time delays, PMBGNP$_M$ can create flexible programs for different dynamic environments.

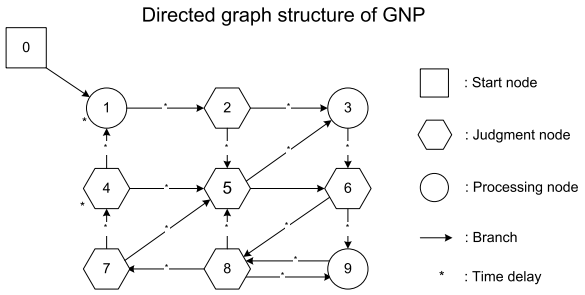

Fig. 1. Basic structure of PMBGNP$_M$.

B. Flowchart of PMBGNP$_M$

Fig. 2 shows the flowchart of PMBGNP$_M$. In PMBGNP$_M$, there are several populations, i.e., R. Each population consists of a number of individuals, i.e., M. All the individuals will be initialized by random and be evaluated by a predefined fitness function. With their fitness values, the elite individuals would be selected. For each population, PMBGNP$_M$ constructs a probabilistic model by estimating the probability distribution from the elite individuals. The probabilistic models are represented by connection probability vectors. Therefore, PMBGNP$_M$ consists of $|R|$ probability vectors, which are denoted as P_r ($r \in R$). Comparing with conventional PMBGNP which constructs single probability vector, PMBGNP$_M$ has more chance to avoid the local optimal convergence with small population size.

On the other hand, with a small population size, the bootstrap problem[1] will be significantly emerged in Evolutionary Robotics [13]. Therefore, if the initial population has poor performances, the outstanding individuals would hardly be generated by PMBGNP. One possible solution to overcome this problem is to simply increase the population size, however, which will be with the increasing cost of computational time. In this paper, another solution is proposed by applying genetic

[1]The bootstrap problem is often described as that: if all individuals from the first initialized population perform equally poorly, the evolutionary process would hardly generate interesting solutions.

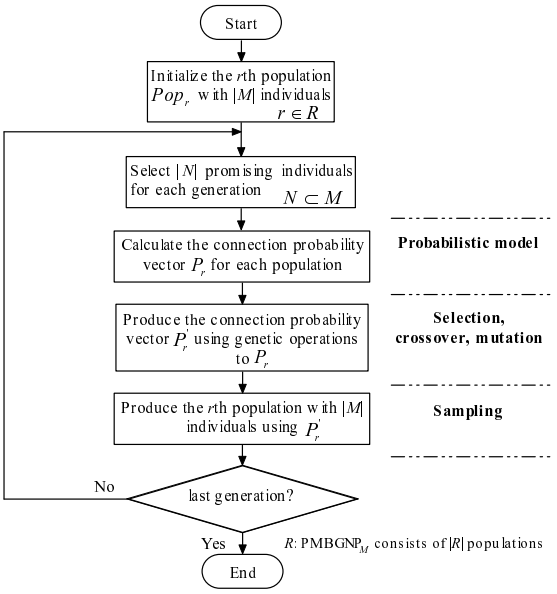

Fig. 2. Flowchart of PMBGNP$_M$.

operations to the probability vectors, which will not only maintain and improve the diversity of population, but also keep the small population size. The selection, crossover and mutation are applied to the constructed probability vectors P_r to produce new probability vectors $P_r^{'}$.

The new $|R|$ populations will be generated by sampling the probability vectors $P_r^{'}$. Crossover and mutation are used to generate the new population in GNP, as a result, the strongly related sub-structures sometimes will be broken down to produce uninteresting individuals, while the probability model is carried out by learning the structure of promising individuals to guide the evolution in PMBGNP$_M$. With the learning of promising individuals, the *Building Blocks* could be recognized and represented implicitly in the probabilistic model, and the generated population becomes capable of inheriting the characteristics of previous good individuals, which would avoid the breakage of *Building Blocks* and improve the exploitation ability. In the next subsections, the construction of probabilistic model, implementation of genetic operations and production of new populations will be introduced.

C. Probabilistic model

For a population r ($r \in R$), the probabilistic model of population r is represented as a probability vector P_r. The connection probabilities between different nodes are considered to construct the probabilistic model of PMBGNP$_M$. The probability vectors are used by learning the structure of promising individuals. For the r_{th} population, P_r^t denotes the probability vector in the t_{th} generation, and $P_r^t(i,k,j)$ represents the connection probability from the k_{th} branch of node i to node j. The probability vector P_r^t is composed of a set of $P_r^t(i,k,j)$.

Fig. 3 shows the procedure to carry out the probabilistic model. PMBGNP$_M$ selects a set of elite individuals from the

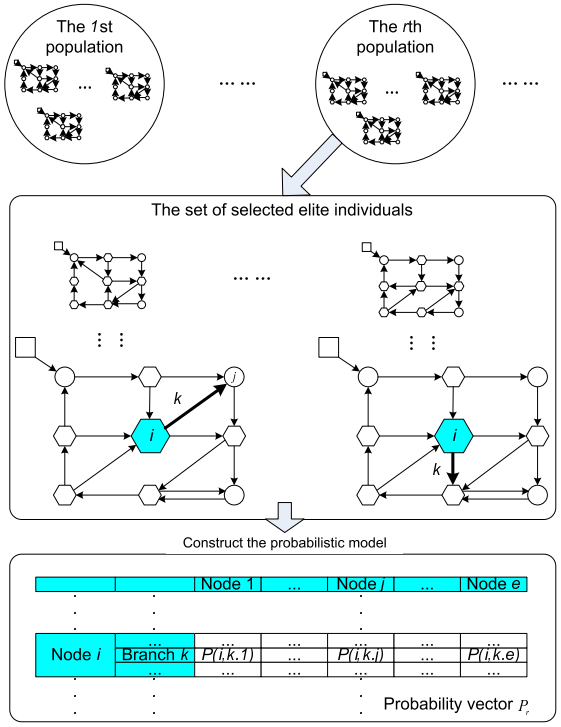

Fig. 3. The procedure of PMBGNP$_M$ to construct the probabilistic model.

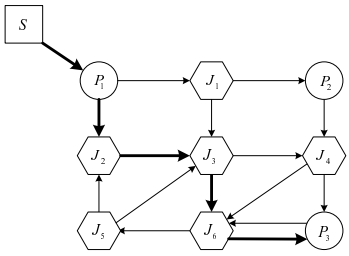

Fig. 4. An example of node transition.

r_{th} population. The probability vector P_r consists of all the structure information of the elite individuals. For example, the information of the k_{th} branch of node i is used to obtain various probabilities to connect to the next node, such as $P(i,k,1)$, $P(i,k,2)$,... etc.

These connection probabilities are calculated by considering the connection information and transition information between different nodes. The reason why we consider these two factors is as follows. In GNP, the directed graph structure is based on the connections of different nodes by branches. Therefore, the connection information is the most important information which deserves to be studied. On the other hand, in GNP, usually not all the nodes are used for transition to solve the problems in one individual. The node transition is made by selecting the necessary nodes. In an example shown in Fig. 4, the nodes are transited like $P_1 \rightarrow J_2 \rightarrow J_3 \rightarrow J_6 \rightarrow P_3$ to solve the problems, which means the information on the connections between these transited nodes is useful for solving the problems.

Therefore, for the r_{th} population, the probability vector in the t_{th} generation is calculated as follows.

$$P_r^t(i,k,j) = \frac{\sum_{n \in N} (\delta_n^t(i,k,j) + \eta \sigma_n^t(i,k,j))}{\sum_{j* \in A(i,k)} \sum_{n \in N} (\delta_n^t(i,k,j*) + \eta \sigma_n^t(i,k,j*))}, \quad (1)$$

where,
N: set of suffixes of elite individuals.
$A(i,k)$: set of suffixes of connected nodes from the k_{th} branch of node i.
$\delta_n^t(i,k,j)$: value defined by

$$\delta_n^t(i,k,j) = \begin{cases} 1 & \text{if the } k_{th} \text{ branch of node } i \text{ in individual } n \text{ is connected to node } j \text{ in the } t_{th} \text{ generation,} \\ 0 & \text{otherwise.} \end{cases}$$

η: coefficient.
$\sigma_n^t(i,k,j)$: value defined by

$\sigma_n^t(i,k,j) = \ell$ if the transition from the k_{th} branch of node i to node j in individual n occurs ℓ times in the t_{th} generation.

Besides, the following exponential smoothing method is considered to update the current probabilistic model considering the previous generation's probability vectors.

$$P_r(i,k,j) \leftarrow (1-\alpha) P_r(i,k,j) + \alpha P_r^t(i,k,j), \quad (2)$$

Here, the coefficient α can be considered as a smoothing rate and $\alpha \in (0,1)$.

After modeling the probability distribution, the probabilistic model consists of $|R|$ probability vectors with regard to all the populations, which could be denoted as follows.

$$P = \{P_r | r \in R\}, \quad (3)$$

$$P_r = \{P_r(i,k,j) | i \in I, k \in A(i), j \in A(i,k)\}, \quad (4)$$

where,
R: set of suffixes of populations.
I: set of suffixes of nodes.
$A(i)$: set of suffixes of branches of node i.

D. Genetic operations

As described in the previous subsections, genetic operations are applied to evolve the constructed probabilistic model P. Selection, crossover and mutation are designed to produce the new probabilistic model $P^{'}$.

The role of genetic operations to the probabilistic model is to change the probability vectors. In each generation, the constructed multiple probability vectors P_r are replaced with the new ones generated by crossover and mutation. Tournament selection is used in PMBGNP$_M$ to select probability vectors for crossover and mutation. Crossover and mutation operations are carried out subject to the following condition.

$$\sum_{j \in A(i,k)} P_r(i,k,j) = 1, \quad (5)$$

1) Crossover: Crossover is executed between two probability vectors and produces two new probability vectors. Crossover operation exchanges all the probability values of the selected branches, as shown in Algorithm 1. The crossover operation satisfies the condition shown in Eq. (5) directly.

Algorithm 1 Crossover of PMBGNP$_M$

1: $m, n \in R$
Select two probability vectors P_m and P_n from P.
2: Each branch (i,k) is selected as a crossover branch with the probability of $p_{crossover}$.
3: Two probability vectors exchange the probability values of the corresponding crossover branches (i.e., $P_m(i,k,j)$ and $P_n(i,k,j)$ are exchanged to form new probability values $P_m^{'}(i,k,j)$ and $P_n^{'}(i,k,j)$).

2) Mutation: Mutation is executed in one probability vector to produce a new one, which is similar to the conventional mutation in evolutionary algorithms. The probability values of the selected mutation branches are changed randomly by mutation operation, where such procedure should satisfy the condition shown in Eq. (5). In order to satisfy this condition, the procedure of mutation is designed as follows.

1) Select one probability vector P_r from P.
2) Each branch (i,k) is selected as a mutation branch with the probability of $p_{mutation}$.
3) For each node j ($j \in A(i,k)$) randomly generate a value $m(j)$.
4) Generate new probability vector $P_r^{'}$ using the following Equation.
$P_r^{'}(i,k,j) = \frac{m(j)}{\sum_{j \in A(i,k)} m(j)}$.

E. Generation of new populations

After the previous steps, the probabilistic model denoted as $P^{'}$ has been produced, which consists of multiple probability vectors $P_r^{'}$. The constructed probability vectors is sampled to generate the new populations as shown in Algorithm 2.

Algorithm 2 Generating a new individual using probability vector $P_r^{'}$ in PMBGNP$_M$

1: set all branches of all nodes as unprocessed
$i \leftarrow 0$, $k \leftarrow 0$
2: decide node j which is connected from the k_{th} branch of node i with probability $P_r^{'}(i,k,j)$
3: $k \leftarrow k+1$
if there are unprocessed branches left in node i, go to 2, otherwise go to 4
4: $i \leftarrow i+1$, $k \leftarrow 0$
if there are unprocessed nodes left, go to 2, otherwise end the generation of a new individual

III. Simulations

A comparative study of the the proposed algorithm with GNP and PMBGNP is carried out in this section. The simu-

lations are designed to control a kind of autonomous robots, Khepera robot [12].

A. Settings of the simulations

1) Settings of the robot: The simulated Khepera robot includes 8 infrared sensors allowing it to detect the proximity of objects in front of it, behind it, and to the right and left sides of it by reflexion. Each sensor returns a value ranging between 0 and 1023[2]. Two motors corresponding to the left wheel and right wheel can take speed values ranging between -10 and +10. Some random noises are added to the amplitude of the motor speed and the direction resulting from the difference of the speeds of the motors (see Fig. 5).

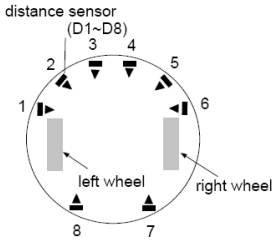

Fig. 5. Khepera robot.

2) Wall following problem: The proposed algorithm is evaluated by solving wall following problem of Khepera Robot. Fig. 6 shows the environment used in this simulation. The size of the simulated environment is 1 m × 1 m including obstacles (walls) in it. The task ends when the time step reaches a predefined steps (1000 in this paper). The wall following behavior of robot is evolved by evolution. Reward and fitness are designed to make the robot move along the wall as fast as and as straight as possible, as shown in the following.

$$Reward = \frac{v_R + v_L}{20} \times (1 - \sqrt{\frac{|v_R + v_L|}{20}}) \times C, \tag{6}$$

$$Fitness = \frac{(\sum_{step=1}^{1000} Reward)}{1000}, \tag{7}$$

where,

v_R, v_L: the speed of right and left wheels.

$$C = \begin{cases} 1 & \text{at least one of the sensor values} \\ & \text{is more than 1000, and the other} \\ & \text{sensor values are less than 100,} \\ 0 & \text{otherwise.} \end{cases}$$

3) Settings of GNP, PMBGNP and PMBGNP$_M$: The node functions used for Khepera robot are shown in Table. I. Each judgment node returns A or B (In this paper, we set the number of branches of judgment nodes at 2.) as a judgment result to determine the robot's next action. Each processing node determines the speed of the left or right wheel. The parameter settings are shown in Table II. In order to study the proposed algorithm, we set the number of populations at six, not single one unlike conventional algorithms. And the total number of individuals is set at small to evaluate the motivation of this work. The crossover and mutation probabilities in GNP and PMBGNP$_M$ are set appropriately through the simulations, i. e., $p_{crossover} = 0.1$ and $p_{mutation} = 0.01$.

[2]0 means that no boject is perceived while 1023 means that an object is very close to the sensor (almost touching the sensor).

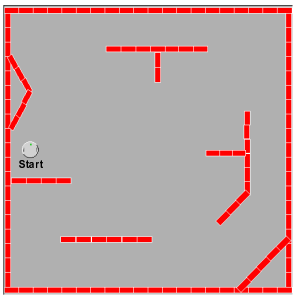

Fig. 6. Simulation environment.

TABLE I
NODE FUNCTIONS USED FOR KHEPERA ROBOT.

Node	Function
J_1, J_2, ..., J_8	Judge the value of the sensor 1, 2, ..., 8
P_1	Determine the speed of the right wheel
P_2	Determine the speed of the left wheel

B. Simulation results

Fig. 7 shows the average fitness curves of the best individuals in each generation over 30 independent simulations. In order to confirm the effectiveness of the proposed algorithm, it is compared with GNP and PMBGNP. It is found that PMBGNP and PMBGNP$_M$ show better fitness at the beginning of the generations because the probabilistic models have higher evolution ability to find better solutions. When the generation goes on, PMBGNP achieves the worst results. The reason is that when the population size is set at small, the diversity of population cannot be guaranteed, which makes the the robot converge to local optima, and causes the bootstrap problem in robots. GNP shows better fitness than PMBGNP because even if poor individuals are generated in the initial generation, crossover and mutation can maintain the diversity in each generation that avoids the premature convergence. PMBGNP$_M$ shows the best fitness among the three algorithms. Comparing with PMBGNP, the multiple probability vectors and genetic operations of PMBGNP$_M$ can maintain the diversity of populations that avoids the local convergence. On the other hand,

TABLE II
PARAMETER SETTINGS.

Parameter	GNP	PMBGNP	PMBGNP$_M$
Number of populations	1	1	6
Number of individuals per population	–	–	50
Total number of individuals	300		
Number of promising individuals	–	150	25
the number of nodes	40(judgment node), 20(processing node)		

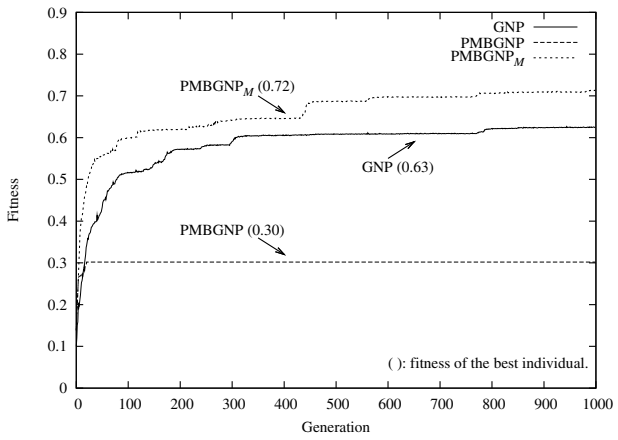

Fig. 7. Simulation results of GNP, PMBGNP and PMBGNP$_M$.

PMBGNP$_M$ inherits the advantages of probabilistic model building evolutionary algorithms. A probabilistic model is constructed by learning the promising individuals that guide the more effective evolution than GNP. The simulation results shows that the proposed algorithm PMBGNP$_M$ can find a better trade-off between exploitation and exploration, comparing with GNP and PMBGNP.

Fig. 8 shows a successful track of the best solution controlled by PMBGNP$_M$. The figure shows that the proposed algorithm can solve the wall following problem well. The robot can move straight along the wall and avoid the obstacles. In some trails, GNP can also obtain good track. But in others, good results cannot be obtained, thus GNP achieves lower average fitness than PMBGNP$_M$. On the other hand, PMBGNP cannot solve the problem due to the premature convergence.

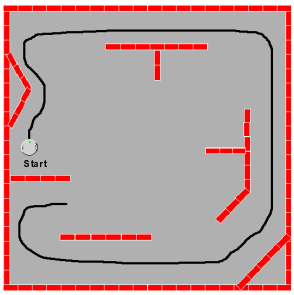

Fig. 8. The successful track of robot by PMBGNP$_M$.

Table III shows the results of t-test (one side) of the mean fitness values between GNP and PMBGNP$_M$. The p value shows that there is a significant difference between GNP and PMBGNP$_M$.

IV. Conclusion

In this paper, an evolutionary algorithm named Probabilistic Model Building Genetic Network Programming using Multiple Probability Vectors (PMBGNP$_M$) has been proposed and applied to control the movement of autonomous robots. PMBGNP$_M$ is an extension of PMBGNP which combines GNP and EDA to improve the evolution ability. There are some advantages of the proposed algorithm. Firstly, comparing with the conventional EDA algorithms, PMBGNP$_M$ can deal with the dynamic environments more effectively and efficiently, due to the directed graph based structures. On the other hand, comparing with the GNP and PMBGNP, the proposed algorithm can solve problems even if the population size is set at small. It is found from the simulations that the higher fitness can be obtained by the proposed algorithm, because multiple probability vectors and genetic operations are combined with the probabilistic model to maintain the diversity of populations, which avoids the local convergence. The proposed algorithm can obtain a better trade-off between exploitation and exploration.

TABLE III
Result of t-test between GNP and PMBGNP$_M$.

	GNP	PMBGNP$_M$
Mean	0.63	0.72
Standard deviation	0.088	0.058
T-test (p value)	0.0074	–

References

[1] K. Hirasawa, M. Okubo, H. Katagiri, J. Hu and J. Murata, "Comparison between Genetic Network Programming (GNP) and Genetic Programming (GP)", *In Proc. of IEEE Congress on Evolutionary Computation*, pp. 1276-1282, 2001.

[2] T. Eguchi, K. Hirasawa, J. Hu and N. Ota, "A Study of Evolutionary Multiagent Models Based on Symbiosis", *IEEE Trans. on Systems, Man and Cybernetics*, Part B, Vol. 36, No. 1, pp. 179-193, 2006.

[3] S. Mabu, K. Hirasawa and J. Hu, "A Graph-Based Evolutionary Algorithm: Genetic Network Programming (GNP) and Its Extension Using Reinforcement Learning", *Evolutionary Computation*, MIT Press, Vol. 15, No. 3, pp. 369-398, 2007.

[4] D. E. Goldberg, "Genetic Algorithm in Search, Optimization and Machine Learning", Addison-Wesley, 1989.

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

[6] D. B. Fogel, "An Introduction to Simulated Evolutionary Optimization", *IEEE Trans. on Neural Networks*, Vol. 5, No. 1, pp. 3-14, 1994.

[7] 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 ICROS-SICE Int'l Conf.*, pp. 3457-3462, 2009.

[8] X. Li, S. Mabu, H. Zhou, K. Shimada and K. Hirasawa, "Genetic Network Programming with Estimation of Distribution Algorithms for Class Association Rule Mining in Traffic Prediction", *in Proc. of the IEEE Congress on Evolutionary Computation (CEC 2010)*, pp. 2673-2680, 2010.

[9] P. Larrañaga and J. A. Lozano, "Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation", Kluwer Academic Publishers, 2002.

[10] M. Pelikan, D. E. Goldberg and E. Cantu-Paz, "Linkage Problem, Distribution Estimation, and Bayesian Networks", Evolutionary Computation, Vol.8, No.3, pp. 311-341, 2002.

[11] H. Handa, "The Effectiveness of Mutation Operation in the case of Estimation of Distribution Algorithms", *BioSystems*, (87)243-251, 2007.

[12] O. Michel, "Khepera Simulator Package Version 2.0: Freeware Mobile Robot Simulator Written at the Universty of Nice Sophia-Antipolis by Olivier Michel", Downloadable from the World Wide Web at http://diwww.epfl.ch/lami/team/michel/khep-sim/.

[13] S. Nolfi and D. Floreano, "Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines", Bradford Book, 2000.