Book Title
Book Editors
IOS Press, 2003

An Update Method of Computer Simulation for Evolutionary Robotics

Yoshiaki KATADA [a,1] and Kazuhiro OHKURA [b]
[a] *Faculty of Engineering, Setsunan University*
[b] *Faculty of Engineering, Kobe University*

Abstract. One of advantages of evolutionary robotics over other approaches is its parallel population search. However, it generally takes an unrealistically long time to evaluate all candidate solutions by using a real robot. Thus, a technique of computer simulation is considered to be one of the most important topics in evolutionary robotics. Although it is quite difficult to provide a precise computer simulation model of a physical experiment, simulated robot/environment interaction dynamics must be synchronously consistent with the physical one during its evolution for reducing the gap between the simulated and physical one. In this paper, in order to overcome this problem, we propose an update method of simulated robot/environment interaction dynamics based on a statistical test during evolution, and then investigate the consistency between the performances of a robot in the simulated and physical environment. A series of experiments with a mobile robot illustrate the effectiveness of the proposed method in practice.

Keywords., Simulation, Real robot, Evolutionary robotics, Embodied cognitive science

1. Introduction

Embodied cognitive science[1] have attracted much research interest in recent years, where a robot must interact with a real world environment for obtaining successful and robust behaviors. One of approaches in embodied cognitive science is evolutionary robotics (ER)[2], where robot control systems are designed by using evolutionary techniques.

It has been pointed out that evolutionary approaches are potentially advantageous over other approaches due to their parallel population search. However, a number of trials needed to evaluate all individual in a real environment make difficult the use of physical robots during evolution, especially in the case of tasks with human intervention. One approach to overcoming this problem would be multiplying the number of the same real robots for the evaluation of individuals [3]. However, a controller should be robust enough to work well for all robots because it may perform differently due to slight differences in the electronics and mechanics of robots, even if they are apparently identical. Addition to this, the approach may result significantly expensive due to preparing a number of robots.

[1]Correspondence to: Yoshiaki KATADA, 17-8 Ikeda-nakamachi, Neyagawa, Osaka 572-8508, JAPAN. Tel./FAX: +81 728 39 9148; E-mail: katada@ele.setsunan.ac.jp

The other approach would be the use of simulations. Computer simulations may be helpful to reduce the amount of experimental time to evaluate individuals in a population. However, the controllers evolved in a simulated environment do not always work well in the real one because of uncertain effects, e.g., noise and differences in the electronics and mechanics of robots, as Brooks pointed out [4]. Therefore, the validity of simulations is particularly relevant problem. Some researchers claimed that the problem described above can be overcome by carefully designing simulators. Jakobi *et al.* claimed that the addition of noise to relevant parts for a control task in a simulated environment can reduce the discrepancy between the simulated and real environment. However, such relevant parts are different on tasks and the detections are crucially dependent on prior knowledges of an experimenter. Miglino *et al.* [6] developed a simulator which requires experimental and multiple samplings for constructing a detailed model of sensors and motors, and claimed that such a method can reduce the performance gap between simulated and real environments. The authors also claimed that the decrease in performance after the transfer of a system from a simulated environment to a real one can be recovered by continuing the evolutionary process in the real environment. However, the method can not adapt to unpredictable changes in a real environment during evolution due to fixed conditions of the simulation. Moreover, performances would not be always improved after the transfer of a system from a simulated environment to the real one without any consideration. An approach to overcoming this problem is to calibrate the model of robot/environment interaction dynamics during robot's life time. Such an approach is found in [7], where a world model is learned during robots' life time. But it is deterministic and composed of only the most recent data set of sensory inputs and motor outputs.

In this paper, we propose an update method of a probabilistic model of robot/environment interaction dynamics for a mobile robot during its evolution employing a statistical test in order to maintain the validity of the model in simulations. The paper is organized as follows. Section 2 describes the procedure of the proposed method. Section 3 defines the robot control problem where the evolved neural networks are evaluated, and representation of robot/environment interaction dynamics. Section 4 gives the results of our experiments. Section 5 discusses the effect of the update method on performances of a robot after the transfer to a real environment. Conclusions are given in the last section.

2. An Update Method of a Probabilistic Model of Robot/Environment Interaction

Simulated and real robot/environment interaction dynamics are denoted $M(s_M, a_M)$ and $R(s_R, a_R)$ respectively, where s is sensory inputs and a is actions. The procedure of evolutionary algorithms with an update method can be summarized as follows:

0. Create an initial population and initial model, $M_0(s_M, a_M)$.
1. Divide individuals in a population into ones to be evaluated in a real environment and the others in a simulated one.
2. Gather data, $R(s_R, a_R)$, while robots are evaluated in a real environment.
3. Update $M(s_M, a_M)$ based on $R(s_R, a_R)$.
4. Evaluate the individuals in the simulated environment, $M(s_M, a_M)$.

5. Select the individuals according to the fitness values evaluated in the simulated and real environment, then apply genetic operations to create a new population of the same size.
6. Repeat from 1. on until the terminal conditions are satisfied.

In the step 3, the Student t-test is employed in order to examine the validity of a model in a simulation; $R(s_R, a_R)$ and $M(s_M, a_M)$ are defined to be statistically identical if we fail to reject the null hypothesis that they are not significantly different. Otherwise, $M(s_M, a_M)$ is corrected repeatedly until we fail to reject it. Such a definition seems appropriate for a probabilistic model like the proposed method. The details are described in the experimental setup.

3. Experimental Setup

3.1. The Task and the Fitness Function

A two-wheeled robot was used in the experiment (Figure 1). The robot's source of sensory input comes from an omnidirectional camera. The environment of the robot was a rectangular arena 2400×2200 mm surrounded by walls colored black with a red target placed at the upper right corner (Figure 2). The control task used in this paper was a goal reach problem where a robot approaches to a target (goal).

Each individual of a population was tested on the robot 4 times for around 40 seconds each (40 sensory-motor steps). At the beginning of each trial, a robot was always placed at the same initial position, the bottom left corner, at either an upward orientation or a rightward orientation (Figure 2). One trial ends either when the robot reaches the goal or when 40 steps are performed without the goal. The performance measure to be maximized was as follows: $Fitness = 1/NumTrials \sum_{i=1}^{NumTrials}(1 - Step/MaxStep)$ where $NumTrials$ is the number of trials for a robot (2 trials for each initial orientation of a robot) and $MaxStep$ is set at 40. The fitness function increases as the robot reaches the goal more quickly.

3.2. Representation of Robot/Environment Interaction Dynamics

3.2.1. Simulating Motors

The number of the combination of motor commands used for operating the wheels of the robot is 400 (20 motor commands are available for each wheel). Thus, a_R is composed of the set of displacements of a robot for each axis in figure 3, $(X_R(i,j), Y_R(i,j), \Theta_R(i,j))$,

Figure 1. A mobile robot

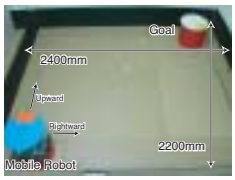


Figure 2. Experimental setup for a goal reach problem

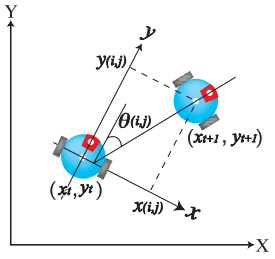


Figure 3. Displacement of a mobile robot in a simulated environment

for a motor command, (i, j) ($i, j = 1, \cdots, 20$ for the left and right wheel). Based on these data, the set of displacements for each axis in a simulated environment, $(X_M(i,j), Y_M(i,j), \Theta_M(i,j))$, is constructed as a_M. In a probabilistic form, the displacements of a two-wheeled robot for each motor command (i, j) are represented as follows [8]: $x(i,j) = N(\mu(X_M(i,j)), \sigma(X_M(i,j)))$, $y(i,j) = N(\mu(Y_M(i,j)), \sigma(Y_M(i,j)))$, $\theta(i,j) = N(\mu(\Theta_M(i,j)), \sigma(\Theta_M(i,j)))$, where $N(\mu, \sigma)$ is the normal distributions with mean, μ, and standard deviation, σ, derived from the sampled data set, $(X_M(i,j), Y_M(i,j), \Theta_M(i,j))$. Thus, a position of a simulated robot at the t-th step for each axis is computed as follows: $x_{t+1} = x_t + x(i,j)$, $y_{t+1} = y_t + y(i,j)$, $\theta_{t+1} = \theta_t + \theta(i,j)$.

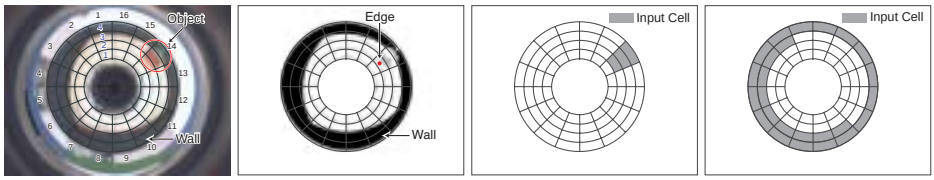
3.2.2. Simulating Sensors

For this experiment, an omni-directional visual sensor was used for the sensory inputs of the robot (Fig. 4). The sensory inputs of an omni-directional vision was computed as follows. The binarized image is taken from an omni-directional image, where the center of the image is identified with the center of a robot (Figure 4(a)). In the image, the direction of the center of gravity of the target to a robot and the distance between a robot and the nearest point of the target are first calculated by using the pixel informations. Next, the image is divided into I individual cells in a circumferential direction and J individual cells in a radial direction (Figure 4(b)) to detect a target by using the above two values (Figure 4(c)) and walls by using the pixel informations with a threshold (Figure 4(d)) on each cell ($q_{ij} = \{0, 1\}, i = 1, \ldots, I; j = 1, \ldots, J$), where q_{ij} signifies the presence or absence of the object. If an object whose height is almost the same as the one of the robot is detected in the j-th cell, the one is also detected in the $(j+1)$-th cell due to the shape of an omni-directional mirror. The walls are also detected for each cell (Figure 4(d)).

In a simulation, omni-directional visual sensors were computed as well as the real ones, assuming binarized images taken from the simulated image. In this experiment, we kept s_M constant during evolution, that is, the model of sensory inputs was not updated due to the reliability of omni-directional visual sensors and the precision of the model based on the characteristic of them.

3.3. An Update Method of Displacements of a Simulated Robot

According to the procedure described in section 2, we explain the details to update and correct a model of displacements of a robot, a_M.



(a) Camera image (b) Binarized image (c) Target detection (d) Wall detection

Figure 4. Omni-directional image plane

For measuring the positions of a real robot during evaluations of it, a CCD camera was positioned on the top of the testing environment. Every 0.5 seconds, the displacements are computed as follows: $x_R(i,j) = x_{t+1}(i,j) - x_t(i,j)$, $y_R(i,j) = y_{t+1}(i,j) - y_t(i,j)$, $\theta_R(i,j) = \theta_{t+1}(i,j) - \theta_t(i,j)$. After the evaluations, all the data set in $(X_R(i,j), Y_R(i,j), \Theta_R(i,j))$ are add to $(X_M(i,j), Y_M(i,j), \Theta_M(i,j))$ as well as the data set before this 500 generations in it are deleted to keep the data set fresh.

In order to examine the validity of a_M, the Student t-test was conducted with the null hypotheses, $\mu(X_M(i,j)) = \mu(X_R(i,j))$, $\mu(Y_M(i,j)) = \mu(Y_R(i,j))$ and $\mu(\Theta_M(i,j)) = \mu(\Theta_R(i,j))$. If the hypotheses are rejected, that is, there are discrepancies between a_M and a_R, then $X_M(i,j), Y_M(i,j), \Theta_M(i,j)$ are corrected, respectively. For instance, if $\mu(X_M(i,j)) < \mu(X_R(i,j))$ and the hypothesis is rejected with a significance level, α, then the smallest data in $X_M(i,j)$ is removed to reduce the discrepancy between them. These corrections are repeated until when all the hypotheses are accepted. Based on the corrected data set, $(X^{'}_M(i,j), Y^{'}_M(i,j), \Theta^{'}_M(i,j))$, displacements of a simulated robot are computed as follows: $x(i,j) = N(\mu(X^{'}_M(i,j)), \sigma(X^{'}_M(i,j)))$, $y(i,j) = N(\mu(Y^{'}_M(i,j)), \sigma(Y^{'}_M(i,j)))$, $\theta(i,j) = N(\mu(\Theta^{'}_M(i,j)), \sigma(\Theta^{'}_M(i,j)))$.

In the experiments carried out in the next section, we take $\alpha = 0.05$. For constructing initial a_{M_0}, 10 data set are sampled for each motor command in advance. a_M is updated after evaluations of real robots equipped with the best and second controller every 50 generations.

3.4. Simulation Conditions

For this experiment, an agent's controller was pulsed neural networks (PNN) with 128 sensory neurons corresponding to the number of cells $I \times J$ in Figure 4(c) and 4(d) for detecting a target and walls, 2 fully interconnected motor neurons and a fully interconnected hidden neuron. The network's connection weights and the firing threshold for each neuron were genetically encoded and evolved. The total number of parameters was 396. The parameters were mapped linearly onto the following ranges: connection weights, $\omega \in [-1.0, 1.0]$, and thresholds, $\theta \in [0.0, 3.9]$. The parameters of the neurons and synapses were set as follows: $\tau_m = 4$, $\tau_s = 10$, $\Delta^{ax} = 2$ for all neurons and all synapses in the network following the recommendations given in [9]. The experiment was conducted using populations of size 50. Each individual was encoded as a binary string with 10 bits for each parameter. Therefore, the total length of the genotype was $L = 3960$. An extended GA, which is called the operon-GA (OGA)[10], were employed

to evolve the PNN parameters [1]. The OGA uses standard bit mutation and five additional genetic operators: *connection*, *division*, *duplication*, *deletion* and *inversion*. The probabilities for genetic operations were set at 0.3 for *connection* and *division*, 0.6 for *duplication* and 0.3 for *deletion* and *inversion*, based on our previous results in [11,12]. The length of the value list in a locus was 6. The per-bit mutation rate, q, was set at $1/L = 0.00025$. Crossover was not used for the OGA, following Nimwegen's suggestion [13]. Tournament selection was adopted. *Elitism* was optionally applied. The tournament size was set at 6 because the OGA prefers high selection pressure. A generational model was used. Each run lasted $2,000$ generations.

The individuals in a population were divided into two groups before evaluations in a real environment as described in section 2; Two individual (the best and second one) of the population to be evaluated in a real environment and the others in a simulated one.

4. Experimental Results

Figure 5 shows the maximum and average fitness at each generation for the OGA. The maximum fitness increased in the early generations without being trapped on local optima although the fluctuations are large in the middle generations.

In order to verify what happens at the level of behaviors, we compared the trajectories of the best evolved individual of $2,000$ generations in the simulated and in the real environment. Figure 6 and 7 show the typical behaviors from the initial orientation of the best evolved robot. In the simulated environment (Figure 6), the robot at the initial rightward orientation approached to the goal turning left to direct toward it. This trajectory matched almost perfectly the one in the real environment (Figure 7). It demonstrates that the proposed update method reduces very significantly the discrepancy between behaviors in the simulated and real environment. Due to space restrictions we only show results for the behavior at the initial rightward orientation. However, the discrepancy between behaviors in the simulated and real environment was also reduced at the initial upward orientation.

5. Discussion

In order to investigate the validity of the proposed update method, we compared the performance of the robot controller evolved in both a simulated and real environment with

[1] In the preliminary runs, it has been confirmed that the OGA outperformed the standard GA in this problem.

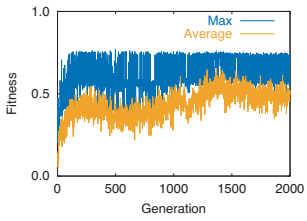


Figure 5. Maximum and average fitness at each generation for the OGA

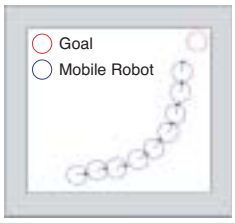


Figure 6. Behavior of the best evolved robot in the simulated environment

Figure 7. Behavior of the best evolved robot in the real environment

Table 1. Success Rate(%) for Each Method

Initial orientation	*Upward*	*Rightward*	*Total*
Without update	10	0	5
With update	40	100	70

the updates, the *method with update*, with the performance of the evolved one only in a simulated environment without any update, the *method without update*. An additional experiment was therefore conducted. We evolved robot controllers only in a simulated environment without any update in $2,000$ generations and then transfered the obtained control system to the real environment. The initial data set, a_{M_0}, were used until the last generation in the method without update. The representation of robot/environment interaction dynamics in the method without update was the same as the one in the method with update. It is likely that the controller evolved only in a simulated environment is not robust enough to work well when transfered to the real environment because it may have discrepancies between the simulated and real environment. Table 1 shows the rates achieving the task in the 10 trials for the controllers evolved by the method with and without update. As predicted, the degradations in performance after the transfer of the system from the simulated environment to the real one are observed in the controllers evolved only in the simulated environment. Moreover, the controller evolved by the method performed better with update than without it for both initial orientation of a robot. This result indicates the effectiveness of the proposed method.

6. Conclusions

In this work, we proposed an update method of simulated robot/environment interaction dynamics for an evolutionary mobile robot during its evolution employing a statistical test in order to maintain the validity of a model in the simulation. Our results can be summarized as follows:

- An update method of a probabilistic model of robot/environment interaction dynamics was designed to reduce the discrepancy between the simulated and real environment.
- The proposed method successfully reduced the gap between performances observed in a simulated environment and performances obtained in the real environment.
- In order to investigate the benefits of the proposed method, the performance of the best evolved robot after the transfer to the real environment is compared with the

one by the method employing only simulation. The proposed method performed better than the method employing only simulation.

These results suggest some guidelines for reducing the gap between performances observed in simulated environment and performances obtained in the real environment for evolved robots. Future work will investigate how well these updating guidelines apply to more complex experiments to be performed, e.g., updating a model of sensory inputs and humanoid robots.

References

[1] R. Pfeifer. and C. Scheier, Understanding Intelligence, MIT Press, Cambridge, 1999.
[2] S. Nolfi and D. Floreano, Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines, MIT Press, 2000.
[3] R. A. Watson, S. G. Ficici and J. B. Pollack, Embodied Evolution: Embodying an Evolutionary Algorithm in a Population of Robots, In Proceedings of Congress on Evolutionary Computation, pp.335-342, 1999.
[4] R. A. Brooks, Artificial Life and Real Robots, In Proceedings of the First European Conference on Artificial Life, pp.3-10, 1992.
[5] N. Jakobi, Half-baked Ad-hoc and Noisy: Minimal Simulation for Evolutionary Robotics, In Proceedings of the Fourth European Conference on Artificial Life, pp.348-357, 1997.
[6] O. Miglino, H. H. Lund and D. Nolfi, Evolving Mobile Robots in Simulated and Real Environments, Artificial Life 2, pp.417-434, 1995.
[7] D. Keymeulen, M. Iwata, K. Konaka, R. Suzuki, Y. Kuniyoshi and T. Higuchi, Off-line Mode-free and On-line Model-based Evolution for Tracking Navigation Using Evolvable Hardware, In Proceedings of the First European Workshop on Evolutionary Robotics, Springer-Verlag, Paris, 1998.
[8] K. Komoriya, E. Oyama and K. Tani, Planning of Landmark Measurement for the Navigation of a Mobile Robot, Journal of the Robotics Society of Japan, Vol. 11, No. 4, pp.533-540, 1993 (in Japanese).
[9] D. Floreano, C. Mattiussi, Evolution of Spiking Neural Controllers, In Gomi, T. (ed.): Evolutionary Robotics: From Intelligent Robots to Artificial Life (ER'01), AAI Books, Springer-Verlag, pp. 38-61, 2001.
[10] K. Ohkura, K, K. Ueda, Adaptation in Dynamic Environment by Using GA with Neutral Mutations, International Journal of Smart Engineering System Design, 2, pp.17-31, 1999.
[11] Y. Katada, K. Ohkura, K. Ueda, Tuning Genetic Algorithms for Problems Including Neutral Networks -A More Complex Case: The Terraced NK Problem-, In Proceedings of the 7th Joint Conference on Information Sciences, pp.1661-1664, 2003.
[12] Y. Katada, K. Ohkura, K. Ueda, An Approach to Evolutionary Robotics Using the Genetic Algorithm with Variable Mutation Rate Strategy, In Proceedings of The 8th Parallel Problem Solving from Nature (PPSN VIII), pp.952-961, 2004.
[13] E. Nimwegen,J. Crutchfield, M. Mitchell, Statistical Dynamics of the Royal Road Genetic Algorithm, Theoretical Computer Science, Vol. 229, No. 1, pp.41-102, 1999