1998 IEEE Asia-Pacific Conference on Circuits and Systems, Chiangmai, Thailand. (IEEE APCCAS 98)

Improving Robustness of Robot Programs Generated by Genetic Programming for Dynamic Environments

Prabhas Chongstitvatana
Department of Computer Engineering
Chulalongkorn University
Phyathai Rd., Bangkok 10330, THAILAND
Tel (662) 218 6956, Fax (662) 218 6955
prabhas@chula.ac.th

Abstract

This paper proposes a method to improve robustness of the robot programs generated by genetic programming. The main idea is to perturb the simulated environment during evolution of the solutions. The resulting robot programs are more robust because they have been evolved to tolerate the changes in their environment. We set out to test this idea using the problem of navigating a mobile robot from a starting point to a target point in an unknown cluttered environment where obstacles can be moved dynamically. The result shows the effectiveness of this scheme.

1. Introduction

The solutions of robot learning problems generated by genetic programming method (GP) are said to be "fragile". That is, they fail to work when there is even a small change in the operational environment such as robots working in the real world. An example can be drawn from our previous work [3], in a visual reaching task, GP is used to generate robot programs that control an arm to reach a target. When there are small changes such as an obstacle is moved from its position, or the control of a real robot misses a step due to random noise, the robot programs fail to work. This is because GP procedure relies on a simulated world to evaluate and to "evolve" the solution. The initial condition for the actual run of a robot program in the real world must be exactly the same as in the simulation, even a small deviation can lead to failure. The accuracy of the world model is an important factor for the success. Most of the work in GP use the simulated world to perform learning. The problem of transferring the result from the simulated world to the real world has been widely recognised [1,2, 5, 7, 10, 11].

To cope with changes, many researchers suggest the use of physical robots to learn in the actual environment of the tasks [5]. The robot will learn by trial and error. This approach is suitable for many learning tasks such as learning the association between sensing and effectors. However, the attempt to use GP as the learning method using this approach will likely take too much time because of the speed limit of a physical robot. The work in [3] shows that for a visual-reaching task, it will take 2,000 hours with their equipments to learn the task. It is possible to reduce the time by running GP in simulation that samples data from the real world [9, 11, 13].

Another approach to cope with changes is to subject the "evolved" system to perturbation expecting that the resulting solution will be more tolerant. The work such as [7, 12] introduce perturbation at every step of evolutionary process. They report limited success. From the experience of our previous work, we notice that we can introduce limited perturbation "in between" generation with good results, i.e. during a generation, we keep everything constant. Another observation is that we can take a successful individual and continue to "evolve" it for a new environment which is changed only slightly from the previous one.

This paper proposes to use perturbation to improve robustness of the robot programs. The main idea is to perturb the simulated environment during evolution of the solutions. The evolution process is carried out such that each individual is evaluated under several environments that are variant of one original. The resulting robot programs are more robust because they have been evolved to tolerate the changes in their environment. We set out to test this idea using the problem of navigating a mobile robot from a starting point to a target point in an unknown cluttered environment (fig. 1). The next section explains the experiment and the result.

2. The Experiments

First, we discuss how to generate a robot program by GP under the normal simulation without perturbation. The mobile robot has capability to move forward and turn left and right. The robot can sense an obstacle in front of it. The environment is simulated on grids of size 600×400 units. The obstacles have several geometrical shapes, each has the average area 20×20 units. The number of obstacles are determined to have their total area 20 % of the whole area. The obstacles are randomly but carefully placed such that they stay some distance from the starting point and the target point to make sure the robot has room to move.

The terminal set in our experiment is { `move, left, right, isnearer?` }. The function set is { `if-and, if-or, if-not` } with the arity 4, 4 and 3 respectively. The `move` returns 1 when the robot hits an obstacle while moving forward and 0 otherwise. The `isnearer?` is the gradient operator to enable the robot to seek the target point. It indicates whether the robot is nearer to the target compared to its position in the previous move.

The fitness measure is based on the distance of the final position of robot to the target and the number of moves. In this experiment smaller value is better. The fitness function is

$$f = kd + m \qquad (1)$$

where d is Euclidean distance of the final position and the target point, $d = 0$ if the robot reaches the target. $k = 10{,}000$ and m = the number of moves. This fitness function promotes the solution with minimum number of moves. The parameters for the genetic programming run are: population size 960, maximum number of generation 200. A simulation run is time-out when a robot executes 10,000 terminals.

The genetic operations are as follows:

1. The population is ranked according to their fitness value and the best 30 are selected as parents.
2. The parents are reproduced to the next generation.
3. All possible pairing of parents are subject to crossover to produce offsprings.
4. The mutation is performed on parents by randomly replacing symbols with a newly generated tree.

The size limit of an offspring is not to be larger than 5 levels tree height.

3. Evolving Robot Programs in a Dynamic Environment

Next, we describe how to use perturbation to improve the robustness of the solutions. To improve robustness of the solutions, the evolution is carried out under several environments each of which is a variation from the original environment. An initial configuration E is perturbed to produce E'. An obstacle is randomly selected and moved in a random direction by a displacement (s). The number of obstacles that is moved is called the percent of disturbance (e).

$$e = \textit{number of obstacles that is moved} \;/\; \textit{total number of obstacles} \qquad (2)$$

The evolution is performed by simulating each individual under a number of environment, say E_n' where n is the number of variation of environment (n = 1 is the original environment). The fitness is evaluated by totaling all the fitness value f_n, where f_n is the fitness under the environment n. During the evolution process, the disturbance is constant. The parameters in the experiments are: s = 8 units, e = 20 %, n = 1, 5, 10, 15, 20.

Robustness are measured by selecting the best individual from the maximum generation and evaluate it under 2000 new environments that are variant of the original. The measure of robustness is the number of success of that individual under these new environments. The percent of disturbance (e) is varied from 0-100% to measure robustness against disturbance.

Result and Discussion

The result is shown in fig. 2. The robustness is better for the larger n, the number of environment. For example, at the disturbance 60 % the robustness improves from 30 % for n = 1 to 80 % for n = 20. The solution evolved under 20 environments (n = 20) shows highest robustness across all the range of disturbance. This result demonstrates clearly the effectiveness of the proposed scheme.

From this result, we have extended our experiment by running the best solution under a continuously changing environment. The environment is perturbed incrementally and continuously every t simulation steps. The initial result confirms that the best solution performs robustly under this environment.

It is noteworthy to observe that many previous work on genetic programming for robot learning are performed with a static environment. The 'dynamic' aspect of the environment is considered a 'disadvantage' that must be dealt with. This work actually 'exploits' this dynamic aspect of the environment and uses it to improve the quality of the solution. Our current activity is concentrated on validating this proposed scheme with the real robot performing in the real world.

References

[1] Chongstitvatana, P. and Polvichai, J. "Learning a visual task by genetic programming", in *Proc. of IEEE/RSJ Int. Conf. on Intelligent robots and systems (IROS96*), Osaka, 1996.

[2] Brooks, R., "Artificial Life to actual robots", in *Proc. of the first European conf. on Artificial Life*, MIT Press, 1991, pp.3-10.

[3] Chang, T., Kuo, S. and Hsu, J., "A two phase navigation system for mobile robots in dynamic environments", in *Proc. of 1994 IEEE/RSJ Int.*

Conf. on Intelligent Robots and Systems, pp. 297-300.

[4] Dorigo, M., “ALECSYS and the AutonoMouse: Learning to control a real robot by distributed classifier systems”, *Machine learning*, vol. 19, 1995, pp.209-240.

[5] Ito, H. Iba and M. Kimura, “Robustness of robot programs generated by Genetic Programming”, in *Proc. of Conf. Genetic Programming 96*, MIT Press, 1996.

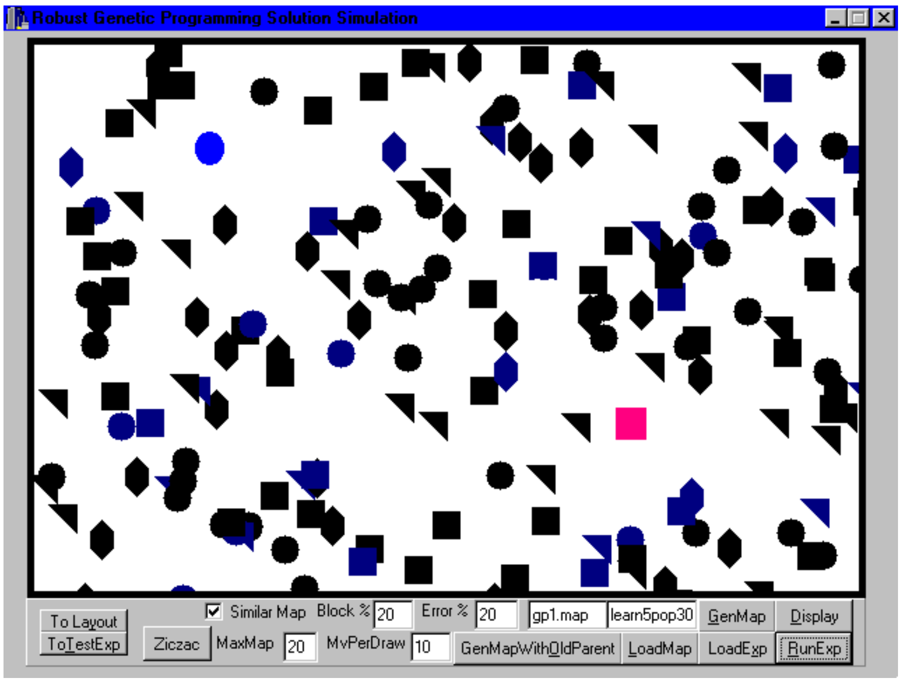
[6] Mataric, M. and Cliff, D., “Challenges in evolving controllers for physical robots”, in *Robotics and autonomous systems*, 19(1):67-83, 1996.

[7] Miglino, O., Lund, H. and Nolfi, S. , “Evolving mobile robots in simulated and real environments”, in *Artificial Life* 2(4), 1996.

[8] Lee, W., Hallam, J. and Lund, H., “Applying genetic programming to evolve behavior primitives and arbitrators for mobile robots”, in *Proc. of IEEE Int. Conf. on Evolutionary Computation*, 1997, pp. 501-506.

[9] Nordin, P. and Banzhaf W., "An on-line method to evolve behavior and to control a miniature robot in real time with genetic programming", in *Adaptive Behavior*, 5(2) : 107-140, 1997.

[10] Reynolds, C., “Evolution of obstacles avoidance behavior: using noise to promote robust solutions”, in K. Kinnear, Ed., *Advances in genetic programming*. MIT Press, 1994.



☑ Similar Map Block % 20 Error % 20 gp1.map learn5pop30

MaxMap 20 MvPerDraw 10

Figure 1 The environment of the mobile robot navigation problem

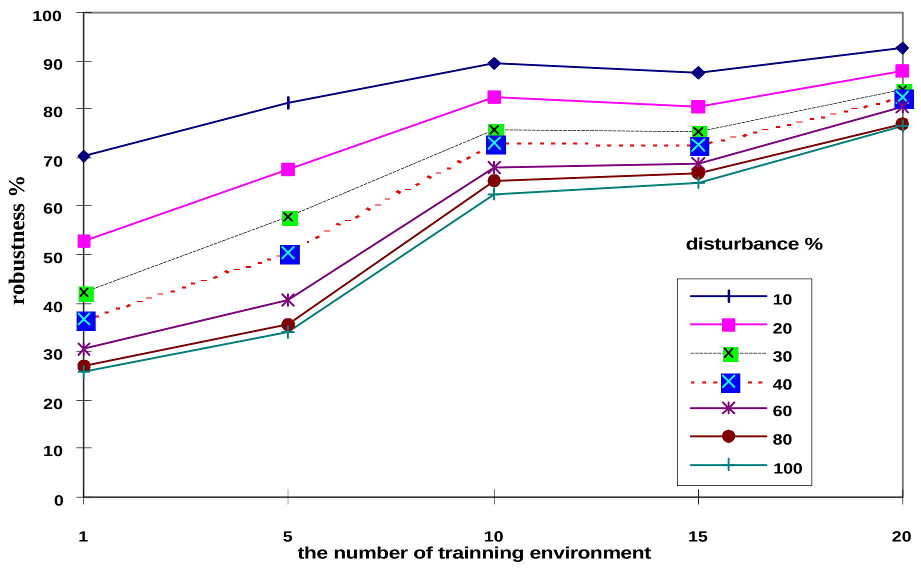


Figure 2 The robustness graph