First Asian Symposium on Industrial Automation and Robotics
BITEC, Bangkok, Thailand
May 6-7,1999

ANALYSIS OF ROBUSTNESS OF ROBOT PROGRAMS GENERATED BY GENETIC PROGRAMMING

Roongroj Nopsuwanchai and Prabhas Chongstitvatana
Department of Computer Engineering, Faculty of Engineering,
Chulalongkorn University, Bangkok 10330, Thailand
Tel: (662)218-6982 Fax: (662)218-6955
prabhas@chula.ac.th

Abstract: *The robot programs generated by Genetic Programming (GP) are found to be 'brittle', i.e. they fail to work when the environment is changed. Perturbation has been used to improve robustness. By introducing perturbation during the evolution of robot programs, the robustness of robot programs can be improved. This paper analyses the cause of the difference of robustness between robot programs using the case of robot navigation problems. The analysis is based on the notion of 'trace' of execution. The result of the analysis shows that the robustness of robot programs depends on the reuse of the 'experience' that a robot program acquired during evolution. To improve robustness, the size of the set of 'experience' should be increased and/or the amount of reusing the 'experience' should be increased.*

Key words: *Automatic Robot Programming, Genetic Programming, Robustness*

1. INTRODUCTION

Our main interest is in the automatic generation of robot programs: given a task description and a particular environment, generate a robot program to perform the task. Genetic Programming (GP) can be used to solve this problem. GP can be regarded as a population based search technique which represents candidate solutions as robot programs. The candidate solutions are said to be evolved until the solution is found. GP uses natural-inspired operators such as selection, reproduction, crossover and mutation operated on candidate solutions to perform the search. It searches a large space before it can find a solution. Therefore, for practical reason, the search is performed in a simulation in which the speed of the robot is not a limiting factor.

Even in the simulated world, the robot programs work successfully only in a particular environment which they were evolved on. They may not work even in that environment if it is slightly changed. It can be seen that the robot programs generated by GP are found to be 'brittle' or lack of robustness. This can be a problem when we transfer robot programs from simulated world to the real world because simulation cannot model the real world exactly. Improving the robustness of robot programs is essential.

In our previous work, Chongstitvatana(1998), GP is used to generated the robot programs that control a mobile robot from a starting point to a target point in a cluttered environment. This is a kind of obstacle avoidance problem. In that work, the robustness of the robot programs is improved by the use of perturbation. The evolutionary process is carried out such that each individual is evaluated under several environments that are the variant of the original one. The result shows that robot programs evolved under such scheme are found to be more robust.

In this work, we investigate the cause of robustness improvement by using the same case as Chongstitvatana (1998). We repeat the experiment with a large number of runs and collect statistical data of robot behavior. Our analysis is based on the notion of 'trace'. A trace is a record of sequence of robot's primitive actions during the execution in an environment. A set of trace in which the robot meets in the training period is called robot's 'experience'. We found that the more robust robot program can reuse its experience during execution in the unseen environment.

he rest of this paper is organized as follows. Section 2 reviews previous work. Section 3 introduces the problems and the experimental set-up. Section 4 elaborates how to improve robustness. The analysis is done in Section 5. Section 6 concludes the paper and discusses future work.

2. RELATED WORK

In this section we discuss some of previous works to demonstrate the use of GP to generate robot programs, the robustness of those programs and the attempt to promote the robustness.

In Chongstitvatana and Polvichai (1996), GP was used to generate robot programs that control a real robot arm avoiding the obstacles to reach the target by visual feedback from the real world . They used the simulation in order to be the learning environment for GP to search for the solutions. Then they transferred the robot programs to perform a task in the real robot arm. The result shows when there are small changes such as an obstacle is moved from its position or the robot misses a step due to random noise in the real world, the robot fails to work even though that robot program performs well and successfully in the simulation. In most cases, the evolutionary process capitalizes on the deterministic, repeatable nature of fitness tests. The individual is repeatedly evaluated in a certain environment. The solution only captures particular characteristics of the environment.

Many researchers propose approaches to increase robustness of the evolved program. Reynolds(1994a) used noise to promote robustness in the obstacle avoidance problem. The robot has to move along the corridor environment without crashing the wall. The noise consists of an error in the robot's input sensor and the output actuator. In this work, Reynolds could not evolve robust controller programs. Later, Reynolds (1994b) changed from using a variable sensor approach to a fixed sensor approach (while maintaining noise). Interestingly, it was this modification that produced more robust solutions. He concludes that GP is capable of evolving robust solutions, and that noise discourages brittle solutions. Another approach to cope with changes is coevolution. Browne (1996) used coevolution technique to coevolve vision-based obstacle avoidance agents. Ito, Iba and Kimura (1996) found that the redundancy of program was effective for generating robust robot program in a box moving problem. Prateeptongkum and Chongstitvatana (1999) examined robustness improvement of robot programs using function set tuning.

Our previous work, Chongstitvatana(1998), in the obstacle avoidance problem, the robustness of robot programs is improved by the use of perturbation. The idea is to perturb the simulated environment during evolution of the solution. Each individual is evaluated under several environments that are perturbed from the original one. We use the same approach in this experiment and analyse its result.

3.THE EXPERIMENTS

This section describes the experiment for generating a robot program by GP under the normal simulation without perturbation. Then we introduce the perturbation to improve the robustness.

3.1 Experimental Set-up

Our problem is to find a robot program that control a robot in a particular cluttered environment from the starting point to the target. The size of the environment as shown in figure 1 is 500× 750 unit. There are many obstacles distributed in this environment. The obstacles have several geometrical shapes, each has the average size of 15×20 unit. The total area of all obstacles is about 20% of the environment whole area. The starting point and the target are fixed in the position as shown. The mobile robot has round shape with its radius 5 unit. It has capability to move forward in the direction of its head, turn left and right. There are sensors for detection the collision with the obstacle, and target determined sensors in both left and right side of its body. Figure 2 show the robot detail.

The terminal set in our experiment is {`forward`, `turnLeft`, `turnRight`, `smellLeft`, `smellRight`}. Each of them activates the robot's primitive action. All of terminals have to return the value after execution. The `forward` moves the robot in the direction of its head by 1 unit, return 1 if it can move successfully and return 0 when it crashes some obstacles or wall. The `turnLeft` and `turnRight` change the robot direction by 22.5° of its previous direction in both left or right orientation

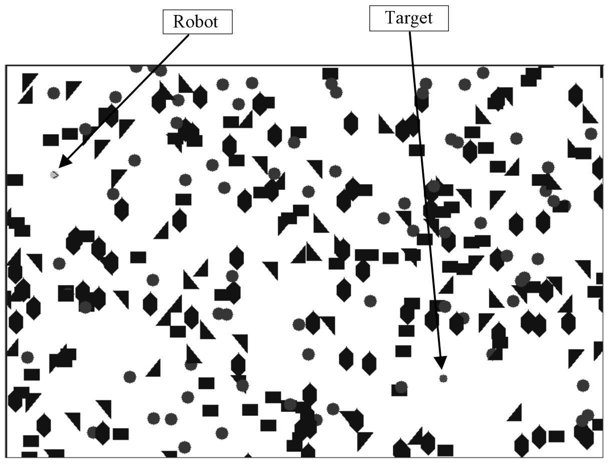

Figure 1. The environment in our experiment

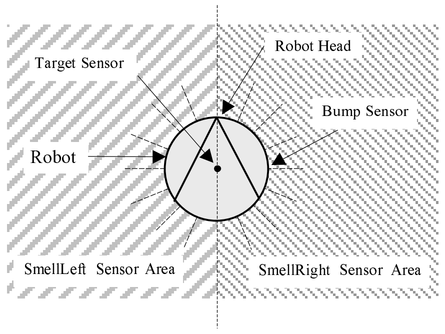

Figure 2. The mobile robot in our experiment

while maintain the previous position, both of these return 1. The `smellLeft` and `smellRight` activate target determined sensor in each side. It determines whether or not there is the target in each sensor area (see figure 2). The return value is 1 if there is the target in that area and is 0 otherwise. The function set we use here is {`IF-AND`, `IF-OR`, `IF-NOT`} with the arity 4, 4 and 3 respectively. It is the basic control flow function that can be used in any problems.

The fitness measurement is based on the distance of the robot's final position and the number of its primitive actions. It is measured after the termination of each individual; when the robot reach the target or it executes more than 5,000 terminals. The fitness function is

$$f = 10{,}000 \times \mathit{finalDist.} + \mathit{numTermPass.} \quad \ldots (1)$$

which *finalDist.* is Euclidean distance of the final position and the target point, *numTermPass.* is the number of executed terminals. The smaller value of fitness is better. The parameters for GP run are shown in table 1. Note that we do not use mutation operator in this experiment. Our crossover operator does not limit the height of the offspring.

3.2 Robustness testing

The robustness of the robot program is defined as an ability of the robot program to perform successfully in the environment that it never meets. In order to measure the robustness of robot program, we create a number of testing environments by perturbing the original environment. From the original environment, we randomly select the obstacle and move it from the original position for 5 units in a random direction. The number of obstacles that is moved is called the percent of disturbance, *d*.

$$d = \frac{\text{number of obstacles that are moved}}{\text{total number of obstacles}} \times 100\% \quad \ldots (2)$$

Table 1. Parameters of GP in the experiment

Population	1,000 programs
initial size of an individual	110 symbols
Maximum generation	125
Reproduction rate	10%
Crossover rate	90%
Mutation rate	0% (do not use)
number of repeated run	20 runs

We create 10,000 testing environments and divide them into 10 groups, each has different of d, call d_{Test} , that varies from 10% to 100% (each group has 1,000 environments). We then select the solution of the experiment as the best individual from the maximum generation, and evaluate it under these groups of environment. The robustness is measured in each group of environment as the percent of number of environment that the robot program can control robot to the target successfully, denoted as $R(d_{Test})$.

4. ROBUSTNESS IMPROVEMENT

In the evolutionary process, the individual is evaluated in one static environment. To improve robustness we introduce perturbation during evolution of the solutions. Each individual is evaluated in a number of environments. A number of training environments are created from one original environment using perturbation similar to the way the tested environments are created. The experiments were performed by varying perturbation in two ways: 1) varying the number of environment during training, keeping the percent of disturbance constant, 2) varying the percent of disturbance d_{Train} during training, keeping the number of environment constant. The individual that works successfully in more environments will have the higher chance to breed the next-generation offspring. An initial environment E is perturbed to produce E'. We produce 49 new environments by perturb the original one E_1', to become E_2' to E_{50}'.

4.1 Varying the number of environment

We use percent of disturbance equal to 10% to create these new environment, d_{Train} = 10%. We set up 7 different experiments that differ in the number of training environments or n (n = 1, 5, 10, 15, 20, 30 and 50).

4.2 Varying the percent of disturbance

The number of environment is 10. The percent of disturbance d_{Trai} is varied 10%, 30% and 50%.

In each experiment, the evolution is performed by evaluating each individual under a number of environments, says E_1 to E_n'. Note that, in the experiment where n = 1, it is the normal simulation without perturbation. The fitness of each individual is evaluated by totaling all the fitness value f_n , where f_n is the fitness under the environment n.

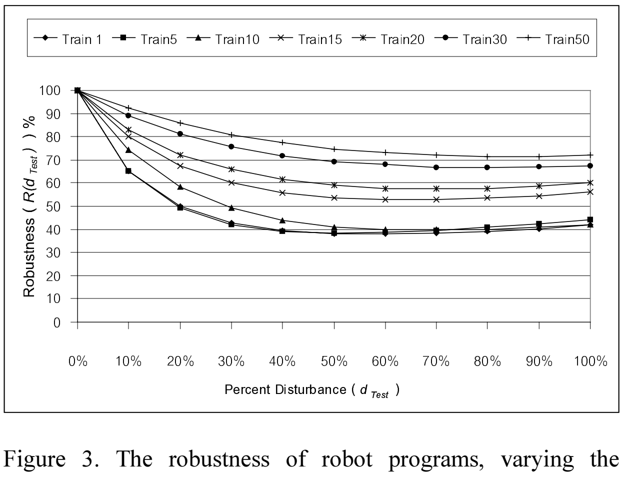

Figure 3. The robustness of robot programs, varying the number of environments

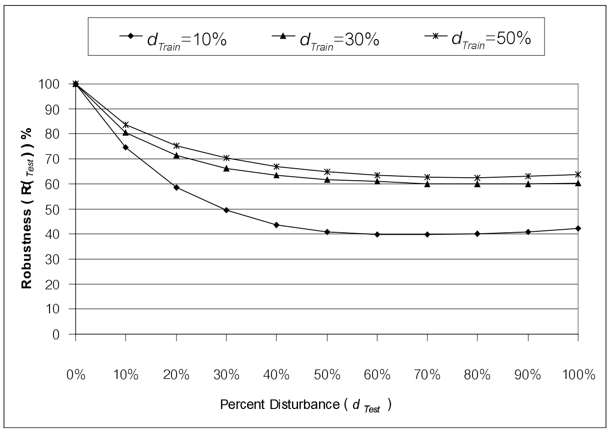

Figure 4. The robustness of robot programs, varying the percent of disturbance

4.3 Result

We run each experiment repeatedly for 20 runs with the different initial population in each run. The robustness of each experiment is compute by average the robustness of the solution from every run. Figure 3 shows the robustness result of the experiment of 4.1 and figure 4 shows the result of the experiment of 4.2. Note that at point d_{Test} = 0%, the robustness of all experiment is 100% because of testing in the original environment. From this result, it is clear that introducing perturbation during the evolution of the solutions can improve the robustness of the solutions.

5. ANALYSIS OF ROBUSTNESS

In this section, we investigate the cause of improvement of robustness by using the result from the experiments. Our analysis is based on the notion of 'trace'.

4.1 Trace

The individual or robot program can be represented in tree form (figure 5). A robot program is evaluated starting from the root through the leaf node. This process is repeated until the termination criteria is met. The execution of a robot program produces a number of sequences of actions. We define this sequence as a 'trace'.

4.2 Analysis of traces

The traces of solution for all runs in both training and testing phases were collected. We define a set L as a collection traces of robot programs that have been evolved under a number of

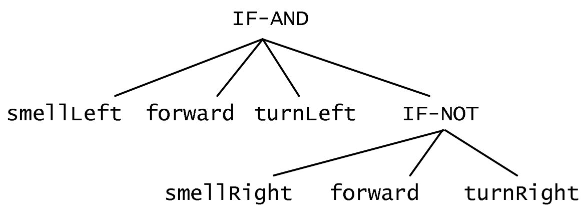

Figure 5. The sample of robot program

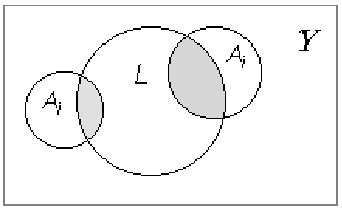

Figure 6. Relationship of set of traces

environments, says n, during training phase. A set L can be regarded as a robot's 'experience' or the response to the different situation that robot has learned. Set L is the subset of all possible traces of that robot program, denoted as Y.

We define a set A_i, a collection of traces when the robot program is executed in testing environments. Figure 6 shows the relation between L and each of A_i. When executing a robot program in an unseen environment, the robot uses its acquired experience. This notion can be captured with the intersection between L and A_i. The amount of the use of the experience of a robot program when executing in i th environment, S_i is defined as

$$S_i = \frac{|L \cap A_i|}{|A_i|} \times 100\% \qquad \text{............(3)}$$

where | | is denoted as the cardinality of the set or the number of element in the set.

The value which we're interested in is the average size of L . We compute the average of S_i for all testing environments. S for the individual that performed successfully is denoted by S_{Succ}, S for the individual that failed is denoted by S_{Fail}. Table 2 shows the result of the experiment 4.1. Table 3 shows the result of the experiment 4.2. Figure 7 shows the relationship between robustness $R(d_{Test})$ and S_{All}. Figure 8 shows the relationship between S_{All} and | L |. It can be seen that S_{All} and | L | are larger after increasing the perturbation level during training. The robustness is improved as a result. A robust solution has a larger set of 'experience' and also reuses them more. From Table 2 and 3, the successful individual has a higher S than the unsuccessful individual ($S_{Succ} > S_{Fail}$ in all setting). The linear correlation coefficient, r, of each curve is shown in the graphs. The conclusion can be made that Robustness $\propto S_{All}$.

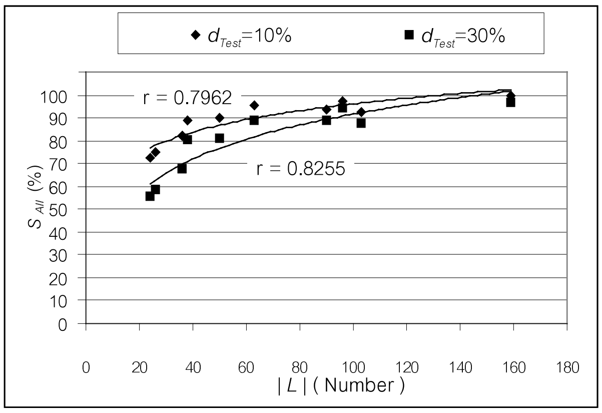

Figure 7. Relationship of robustness and S_{All} of all experiments

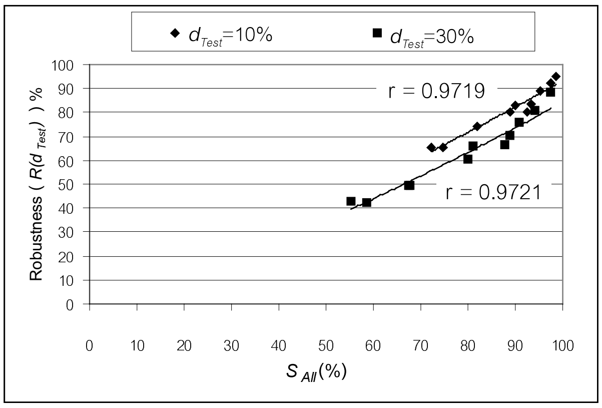

Figure 8. Relationship of | L | and S_{All} of all experiments

Table 2. Analysis of robustness result in each experiment when varies number of training environments

Experiment Result	at d_{Test} =10%				at d_{Test} =30%				| L |
	$R(d_{Test})$	S_{All} %	S_{Succ}%	S_{Fail} %	$R(d_{Test})$	S_{All} %	S_{Succ}%	S_{Fail} %	
Train1	65.19	72.23	83.17	52.61	42.90	55.35	64.19	48.71	24
Train5	65.11	74.77	87.42	52.99	42.15	58.57	69.09	50.88	26
Train10	74.41	82.06	90.77	59.00	49.45	67.75	78.16	57.76	36
Train15	80.17	88.86	93.64	71.26	60.25	80.09	86.29	70.80	38
Train20	82.84	89.89	93.71	72.35	65.85	81.22	85.44	74.21	50
Train30	89.24	95.37	97.09	82.18	75.86	90.86	93.72	82.17	63
Train50	92.52	97.51	98.20	89.59	80.95	94.27	95.85	87.77	96

Table 3. Analysis of robustness result of Train10 experiment when varies d_{Train} 10% 30% and 50%

Experiment Train10	at d_{Test} =10%				at d_{Test} =30%				$\lvert L \rvert$
	$R(d_{Test})$	S_{All} %	S_{Succ}%	S_{Fail} %	$R(d_{Test})$	S_{All} %	S_{Succ}%	S_{Fail} %	
d_{Train}=10%	74.41	82.06	90.77	59.00	49.45	67.75	78.16	57.76	36
d_{Train}=30%	80.37	92.63	94.98	83.81	66.31	87.70	90.33	82.55	103
d_{Train}=50%	83.58	93.40	95.59	82.93	70.22	88.86	91.69	82.37	90

6. CONCLUSION

From our analysis, the robustness of robot program is improved by using perturbation during evolution. Perturbation can be introduced by using a number of training environments and by increasing the perturbation level in each training environment. Robustness is increased because of the larger size of robot's experience and the ability of reusing the experience in the unseen environment.

The analysis of robustness gives many insights on the behavior of GP generated robot programs and on finding more robustness improvement techniques. Our current activity is concentrated on validating this scheme with the real robot performing in the real world.

7. REFERENCES

Browne, D. (1996). "Vision-Based Obstacle Avoidance: A Coevolutionary Approach" *Bachelor Degree Thesis in Department of Software Development,* Monash University, Australia.

Chongstitvatana, P. and J. Polvichai (1996). "Learning a Visual Task by Genetic Programming". In *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and System,* Osaka.

Chongstitvatana, P. (1998). "Improve Robustness of Robot Programs Generated by Genetic Programming for Dynamic Environments". In *Proc. of Asia-Pasific Conference on Circuits and Systems (APCCAS98)*, Chiangmai, Thailand.

Ito, T., H.Iba, and M.Kimura (1996). "Robustness of Robot Programs generated by Genetic Programming". In *Genetic Programming: Proceedings of the First Annual Conference.* MIT Press.

Prateeptongkum, M. and P.Chongstitvatana (1999,March). "Improving the Robustness of a Genetic Programming Learning Method by Function Set Tuning". In *Proc. of Third Annual National Symposium on Computational Science and Engineering (ANSCSE99)*, pp.301-305, Bangkok, Thailand.

Reynolds, C. W. (1994a). "Evolution of Obstacle Avoidance Behavior:Using Noise to Promote Robust Solutions". In K. E. Kinear,Jr. (Ed.), *Advances in Genetic Programming,* Chapter 10, pp.221-241. MIT Press.

Reynolds, C. W. (1994b). "Evolution of Corridor Following Behavior in a Noisy World ". In *Simulation of Adaptative Behavior (SAB-94)*.

Acknowledgements

The first author is supported by the grant from the National Science and Technology Development Agency of Thailand (NSTDA).