SELECTIVE CROSSOVER IN GENETIC PROGRAMMING

Hengpraprohm, S and Chongstitvatana, P

Department of Computer Engineering
Chulalongkorn University
Phayathai Road, Bangkok 10330, Thailand
supojn@yahoo.com, prabhas@chula.ac.th

ABSTRACT

Performance of Genetic Programming depends its genetic operators, especially the crossover operator. The simple crossover randomly swaps subtrees of the parents. The "good" subtree can be destroyed by an inappropriate choice of the crossover point. This work proposes a crossover operator that identifies a good subtree by measuring its impact on the fitness value and recombines good subtrees from parents. The proposed operator, called selective crossover, has been tested on two problems with satisfactory results.

1. INTRODUCTION

Genetic Programming [1] is a search method that imitates natural evolution and natural selection. It is developed from Genetic Algorithms [2] and is differed by the way the solution is represented in a tree structure instead of a fixed length binary string. The candidate solutions formed a population and are evolved via genetic operations. The parents are selected according to their fitness and the crossover operator is applied to produce offspring. The mutation occurs to retain the diversity of the population. The crossover operator is very important in Genetic Programming.

The simple crossover selects the crossover points on two parents randomly (with bias towards the internal nodes) and exchanges their subtrees. The "good" subtree can be destroyed by an inappropriate choice of the crossover point. If the "good" subtree can be identified then it can be protected against the crossover operator. The good subtrees can also be combined to form a fit offspring.

This work proposes a crossover operator that identifies a good subtree by measuring its impact on the fitness value and recombines good subtrees from parents.

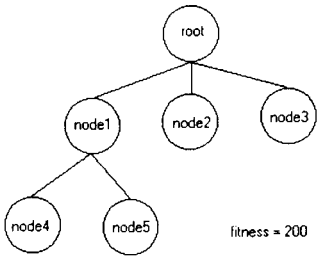
2. RELATED WORK

In a simple crossover there is no way to find out whether the choice of crossover point is good or bad. One way to measure it is to try a number of different choices and choose the best result. This has been proposed by Tackett [3], which he called "brooding selection". Iba [4] suggested to measure the "goodness" of subtrees and use them to bias the choice of crossover points. He proposed a number of estimate using the shape of the tree. Evolutive intron [5] is proposed as a way to protect good subtrees. The main idea is to use the non-action node, so called "intron", to protect the good subtree from the crossover operator.

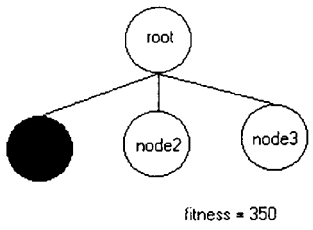
Our method proposes a direct measurement of the impact of subtrees on the fitness value and uses them to choose the crossover points.

3. SELECTIVE CROSSOVER

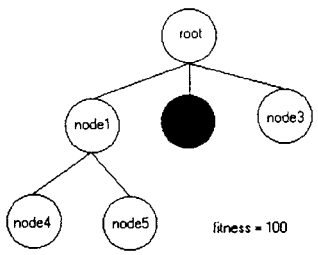
To measure the impact of a subtree on the fitness value, that subtree is pruned from the original tree and the tree is re-evaluated. Pruning is achieved by substituting the subtree by a non-operating node. Two nodes are identified: the best node and the worst node. The best node is the node that has the highest impact on the fitness value, i.e. When it is pruned the fitness value dropped the most (in case we use the maximised fitness function). The worst node is the opposite. It is the node that when it is pruned the fitness value is increased. Figure 1 shows the pruning of 3 nodes: node 1, node 2 and node 5. The fitness value of the original tree is 200. The fitness values after pruning are 350, 100 and 310 respectively. Therefore, the best node is the node 2 and the worst node is the node 1. The best node signifies the good subtree, i.e. without this subtree the fitness of the candidate will be worst. The worst node is the opposite.



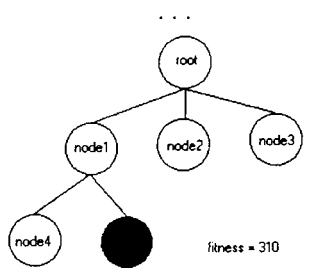
(a) original tree



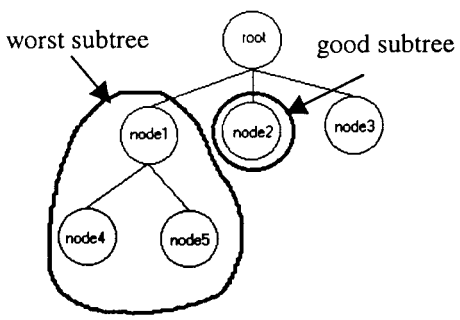
(b) after pruning of node 1



(c) after pruning of node 2



(d) after pruning of node 5



(e) the best and the worst nodes with pruning

Figure 1 Identify the best and the worst nodes with pruning

In the selective crossover, the best node and the worst node are identified by pruning every nodes in the tree. The crossover is performed by substituting the worst node of one parent by the best node of the other parent, therefore combining the good subtrees from both parents to produce offspring (Fig. 2).

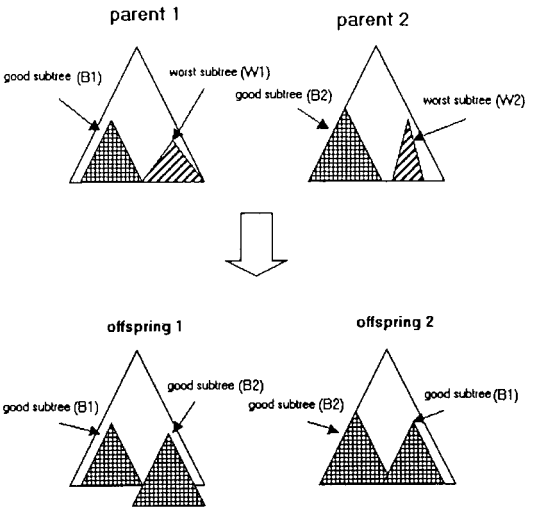


Figure 2 Selective Crossover

4 THE EXPERIMENTS

To test the selective crossover, two problems are chosen: 1) the robot arm control by Jassadapakorn [6] and 2) the artificial ant in Koza [1]. The parameters of GP run are shown in Table 1. The mutation is not used in the experiments.

Table 1 Parameters in GP run

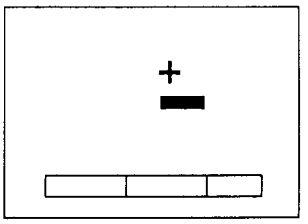
	Robot arm	**Artificial ant**
Population	400	500
Max. generation	10	51
Number of runs	1000	200
Crossover rate	90%	90%
mutation rate	0%	0%

The selective crossover is measured against the simple crossover. The metric is the computational effort as defined by Koza [7]. The computational effort measured the minimum number of candidate that must be processed to find the solution.

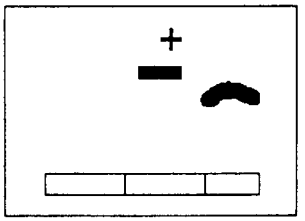
4.1 The robot arm control

The objective of the problem is to generate a program that control a 3 DOF robot arm to reach a

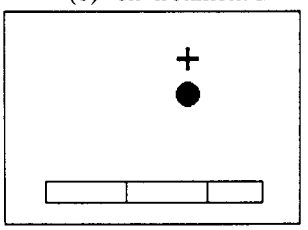
specified target. There are 3 environments contained obstacles. The command primitives are robot joint movements and sensing the collision between the arm and the obstacles.



(a) environment 1



(b) environment 2



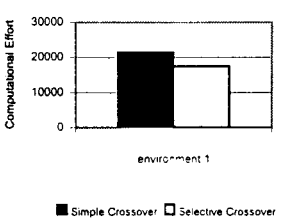
(c) environment 3

Figure 3 The 3 environments in the robot arm control problem

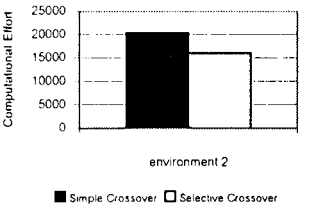
The results are shown in Table 2 and Figure 4. In all cases, the selective crossover achieves a lower computational effort.

Table 2 the results of the robot arm control problem

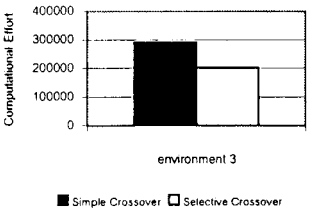
	Environment 1	Environment 2	Environment 3
Simple Crossover	21600	20400	291200
Selective Crossover	17600	16000	202800



(a) environment 1



(b) environment 2



(c) environment 3

Figure 4 Comparison of computational effort of the robot arm control problem

4.2 The Artificial Ant

The objective of this problem is to generate a program that control an artificial ant to find all food tablets in the trail within a limited time. The trail is a 32 by 32 grids contained 89 food tablets. The time limit is when exceeding 400 primitive operations. The trail in the test is called "Santa Fe trail" as defined in Koza [1] and [2] and is shown in Fig. 5.

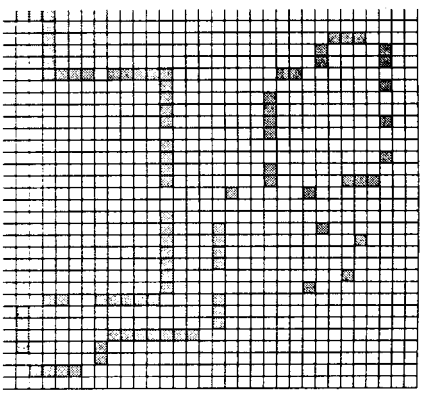


Figure 5 The Santa Fe trail

The result is shown in Table 3 and Figure 6. The selective crossover achieves a lower computational effort.

Table 3 the results of the artificial ant problem

Simple Crossover	Selective Crossover
936000	714000

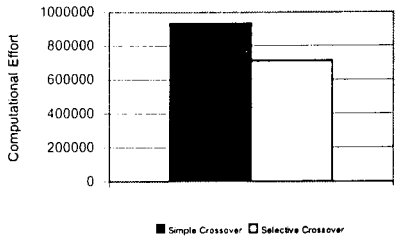


Figure 6 Comparison of computational effort of artificial ant problem

5. CONCLUSIONS

This work demonstrates that using the selective crossover which combines good subtrees from parents and eliminates bad subtrees can achieved a lower computational effort when compared with the simple crossover. The selective crossover uses a direct measurement of the goodness of a subtree by accessing its impact on the fitness value. However, pruning every nodes of a candidate takes a lot of CPU time. In practice, this time consuming computation must be weighted against the gain from faster convergence rate achieved by the selective crossover. The future work will be on reducing the computation time required by pruning.

REFERENCES

[1] Koza, J. , Genetic Programming MIT Press, 1992.

[2] Holland, J., Adaptation in Natural and Artificial System, Ann Arbor, Michigan : University of Michigan Press, 1975.

[3] Tackett, W., Recombination, selection and the gnetic construction of computer programs, PhD thesis, University of Southern California, 1994.

[4] Iba, H, and de Garis, H., "Extending Genetic Programming with Recombinative Guidance", Angeline, P. and Kinnear, K. (eds.), Advances in Genetic Programming Vol 2, MIT Press, 1996.

[5] Carbajal, S, and Martinez, F "Evolutive Introns : A Non – Costly Method of Using Introns in GP", Genetic Programming and Evolvable Machines Vol. 2, 2001. pp. 111 - 122

[6] Jassadapakorn, C. and Chongstitvatana, P., "Reduction of Computational Effort in Genetic Programming by Subroutines", Proc. of National Computer Science and Engineering Conference, NCSEC98, Bangkok, Thailand, 1998.

[7] Koza, J. Genetic Programming II: Automatic Discovery of Reuseable Programs, MIT Press, 1994.