A GP Neutral Function for the Artificial Ant Problem

Esteban Ricalde
IIMAS-UNAM, Circuito Escolar s/n
Ciudad Universitaria
DF, Mexico
ericaldeg@uxmcc2.iimas.unam.mx

Katya Rodríguez-Vázquez
IIMAS-UNAM, Circuito Escolar s/n
Ciudad Universitaria
DF, Mexico
katya@uxdea4.iimas.unam.mx

ABSTRACT

This paper introduces a function that increases the amount of neutrality (inactive code in Genetic Programming) for the Artificial Ant Problem. The objective of this approach is to try to smooth the ridged fitness landscape of the Santa Fe trail.

Several experiments were carried out with different crossover and mutation rates, in order to identify the better settings to solve this problem and to compare the normal representation and the one proposed in this paper. The results indicate that the proposed approach is better than the conventional one.

Also the difference between per individual and per node mutation is showed and a way to relate them is pointed out.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Automatic Programming; D.2.8 [**Software Engineering**]: Metrics - *complexity measures, performance measures*

General Terms

Algorithms

Keywords

Evolutionary Computation, Genetic Programming, Neutrality

1. INTRODUCTION

The Santa Fe trail Artificial Ant Problem [7] is a well studied case often used as a GP benchmark. In [11] Langdon and Poli demonstrated that the classical GP algorithm [9] needs the same number of evaluations than random search to find an optimal solution.

In order to improve the GP performance, diverse modifications to the problem characteristics and to the evolutive process have been made [3, 10, 12, 15, 6].

GECCO '07, July 7–11, 2007, London, England, United Kingdom.

The aim of this paper is to analyze the effects of adding an extra element in the GP function set to allow explicit neutrality. Based on this neutral function, we try to smooth the ridged fitness landscape of the Santa Fe trail Artificial Ant Problem.

The paper is organized as follows. In Section 2, a general perspective of the neutrality researches in GP is presented. Section 3 is a briefly description of the Artificial Ant Problem. Section 4 describes the neutral function introduced in this paper. Section 5 provides details on the used experimental setup and results are presented in section 6. Section 7 discusses these results and in section 8 the conclusions are drawn.

2. NEUTRALITY

A neutral mutation is this that change the genotype without affecting the phenotype. The neutral theory of molecular evolution [8] postulates that in natural systems a considerable fraction of all mutations is neutral with only a small fraction of non-neutral mutations being beneficial.

Today, a lot of work about neutrality in EC exist, some researchers have found it beneficial and others detrimental to the evolutionary process.

In GP, neutrality is often identified with redundancy and introns. Both have been widely studied in the EC community [14, 13, 2].

The problem with functional redundancy and introns is that both emerge and vary during the evolutionary process and for this reason it is difficult to measure neutrality.

Next, a general perspective of the neutrality researches in GP is presented:

In [1], Banzhaf proposed Binary Genetic Programming (BGP) and a genotype-phenotype mapping for some regression problems and concluded that the separation of search operations and the use of a genotype-phenotype mapping gives more flexibility to the evolutive process than working only with phenotypes.

In [12], Miller and Thompson proposed a highly redundant Evolutionary Algorithm called Cartesian Genetic Programming (CGP), they proved it with some landscapes (including the Artificial Ant Problem) and concluded that the CGP is at least equal effective than other forms of GP.

Yu and Miller have reported the use of CGP and Simple Genetic Algorithm (SGA) neutral variations in different landscapes [16, 17, 18] concluding always that neutrality improves the success rate of the evolutive process.

However, Collins [4] analyzed the results of [17] and concluded that, for this problem, the affirmation that neutrality is beneficial is false.

Galván, Rodríguez and Poli [5] analyzed the effects of adding an extra element in the GP function set to allow explicit neutrality in several evolvable hardware problems. They concluded that explicit neutrality has a better overall performance in terms of consistency in reaching feasible solutions.

3. THE ARTIFICIAL ANT PROBLEM

The objective is to find a program to successfully navigate an Artificial Ant along a twisting and discontinuous trail of 89 pieces of food on a 32×32 toroidal grid.

The ant has a reduced set of possible actions: *move one square forward* (and if the new square have food, eat it), *turn right* and *turn left*, each of these operations takes one time unit. It can also *look ahead one square* in the direction it is facing to determine if that square contains a piece of food. This operation is frequently called the sensing function.

In GP, the function set is composed by the *IfFoodAhead* sensing function, which executes one of its two arguments depending upon whether that square contains food or is empty; *Progn2*, which takes two arguments and executes them sequentially and *Progn3*, which do the same that *Progn2* but with three arguments. The terminal set is {*move, right, left*} and each of its elements corresponds to one of the basic ant actions described above.

Only 600 time steps are allocated to execute the ant control program, if a program finishes before 600 time steps then restarts from its root node.

The Artificial Ant must follow the Santa Fe trail, which consists of 144 squares with 21 turns. Usually, the amount of food eaten is used as the fitness measure of the control program, but in this work the fitness function will be the remaining pieces of food of the trail after the program execution (minimize).

4. NEUTRAL FUNCTION

The element to be added to the function set its called *Neut_progn* and has arity 3. Its first argument, called **chooser**, is always a terminal, an integer between 0 and 3. The second and third ones are typical subtrees.

To allow explicit neutrality, depending on the chooser value the behaviour of *Neut_progn* changes. If it is set to 0, none of the other two arguments is executed. When it is set to 1, only the second argument (central edge) is executed. If it is 2, the third argument (right edge) is executed. Finally, if it is set to 3, both (second and third arguments) are sequentially called. Table 1 resumes these actions.

Table 1: *Neut_progn* **operation manner**

Chooser value	Behaviour
0	No argument is executed
1	Only the second argument is executed
2	Only the third argument is executed
3	The second and third arguments are sequentially executed

Figure 1 shows how the amount of inactive code is related to the chooser value. However it is important to emphasize that the minimum change on the chooser value could drastically change the ant behaviour.

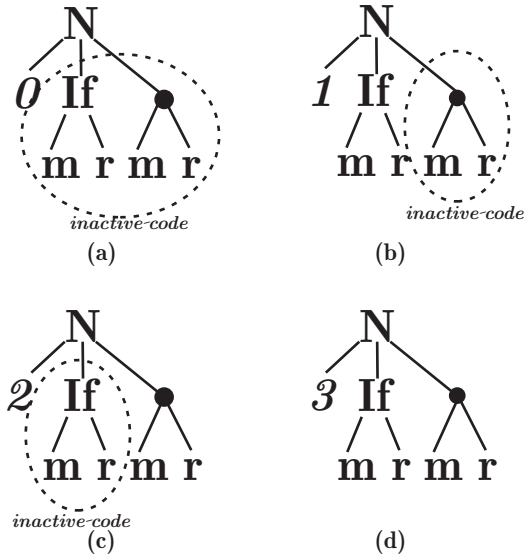

Figure 1: Example of ant control program with the Neut_progn function and influence of the chooser value over the amount of inactive code.

It is clear that when *Neut_progn* is added to the function set of the Artificial Ant Problem, the amount of neutrality increases.

When *Neut_progn* is added, some modifications should be made to the GP process. The crossover point could not be the edge between any *Neut_progn* and its chooser, if this occurs a new crossover point must be selected.

If point mutation [2] is being used with the traditional function and terminal sets for the Artificial Ant Problem, any terminal could be set in place of the others but because of different arities only *Progn2* and *IfFoodAhead* could be interchanged. A mutation in *Progn3* must be ignored. However when *Neut_progn* is added, two approaches arises.

The first one is to ignore *Neut_progn* and *Progn3* mutations (**simple approach**).

The second option (**advanced approach**) is to change *Neut_progn* by *Progn3* function (same arity) and vice versa.

However the left edge must be changed too. In the case of *Neut_progn* to *Progn3* change, the chooser must be changed by a new subtree (expansion mutation). In the case of *Progn3* to *Neut_progn* change, the subtree of the left edge must be deleted and replaced with a randomly chooser value (especial case of a collapse subtree mutation).

Finally, in any case, when a chooser is mutated, a random integer between 0 and 3 (different to the actual chooser value) is set as the mutated chooser value.

5. EXPERIMENTS

Three different series of experiments were performed, in the first one the traditional function set is used. This will be called the **normal** series. The second one includes the *Neut_progn* function with the simple approach for the mu-

Table 2: Summary of GP Parameters

Parameter	Value
Population size	150
Generations	333
Selection	Tournament (size 3)
Initial depth	2 or 3
Final depth	4 or 5
Crossover rate	Variable
Mutation rate (per node)	Variable
Fitness	Remaining food on the trail

tation operator and it is called the **simple neutral** series. The last one includes the *Neut_progn* function, integrating the mutation using the advanced approach and will be called **advanced neutral** series.

After a series of preliminary experiments the population size was fixed to 150, the number of generations to 333, the tournament selection with tournament size of 3 was chose, the initial and final depth were set to 2 and 4 for the normal series and to 3 and 5 for the neutral series (both simple and advanced), point mutation is used and the best individual of each generation directly pass to the next one (elitism).

Fitness is the remaining pellets of food in the trail after the programs execution. Table 2 resumes the parameters settings.

No crossover and mutation rates were fixed because for each series, all the combinations between the crossover rates {20, 45, 60, 85, 95, 100} and mutation rates {0, 1, 2, 3, ..., 11} were mapped.

To obtain meaningful and conclusive results, 50 independent runs were performed in each point.

Two methods were used to measure the performance of the experiments: effort [9] and average of the best fitness value.

The effort is the total number of individuals that need to be processed or the number of program evaluations needed to ensure that an optimal solution has been found. The number of independent runs $R(z)$ required for a probability of success (equation 1) must be solved in order to calculate the effort (equation 2).

$$R(z) = \left\lceil \frac{\log(1-z)}{\log(1-P(M,i))} \right\rceil \quad (1)$$

$$I(M,i,z) = MR(z)i \quad (2)$$

Where z is the success probability, chose as 0.99. M represents the population size, i the generation number and $P(M,i)$ is the cumulative probability of success.

6. RESULTS

Due to space limitations results are presented in a compact format. Figures 2 and 3 show the effort comparison between simple and advanced neutral series with the normal series, respectively. The white squares represent that the neutral version is better than the normal one, black squares mean that normal series is better than neutral one, and finally, gray squares mean that both have the same value.

In both cases the neutral series shows a better performance that the normal one with low mutation rates (between 1 and 7%) and low crossover rates (between 20 and

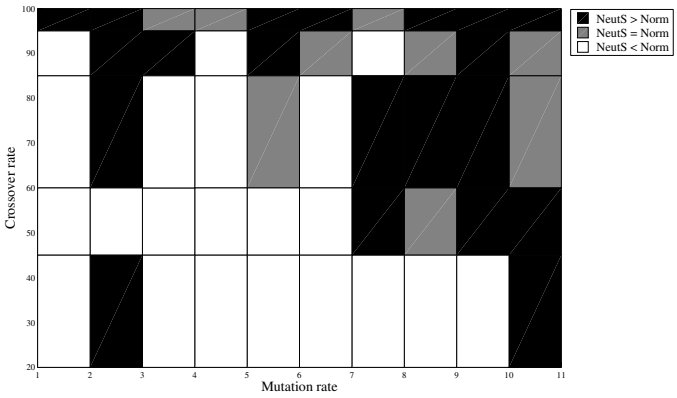

Figure 2: Effort comparison between normal runs and simple neutral ones. Squares are black where normal is better than simple neutral, white where simple neutral is better than normal and gray where they are equal.

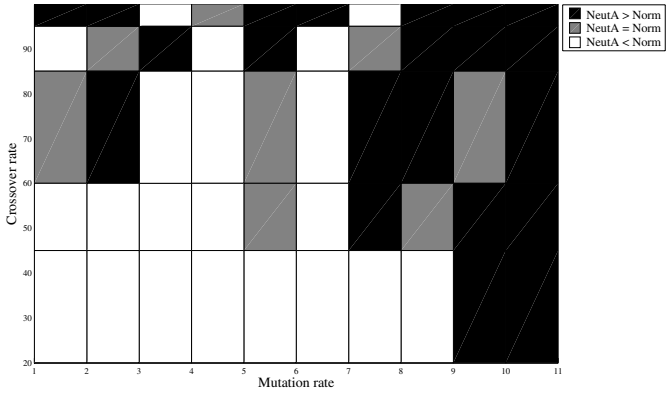

Figure 3: Effort comparison between normal runs and advanced neutral ones. Squares are black where normal is better than advanced neutral, white where advanced neutral is better than normal and gray where they are equal.

60%), with high mutation rates and low crossover rates the normal series seems to be the best option and with high crossover rates it is hard to tell which is the best configuration.

As a detail of the breaking point, figure 4 presents the plot of the effort against the mutation rate of the three series with a crossover rate of 60%.

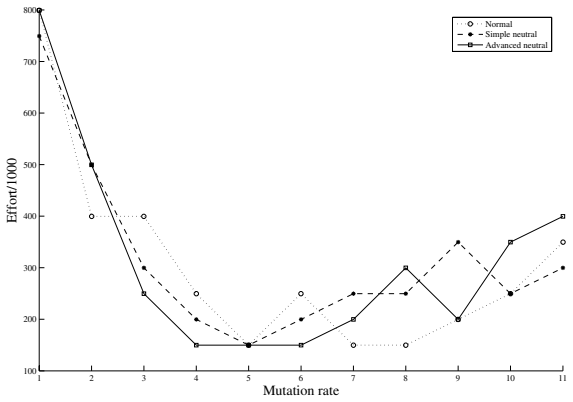

Figure 4: Effort/1000 vs. Mutation rate with a 60% crossover rate for normal, simple and advanced neutral series.

Figures 6 and 7 show the fitness comparison between advanced and simple neutral series with the normal one in a similar way to the used in the effort comparison (fig. 2 and 3) with the difference that gray squares are used when the distance between neutral and normal values is small (less than 0.5).

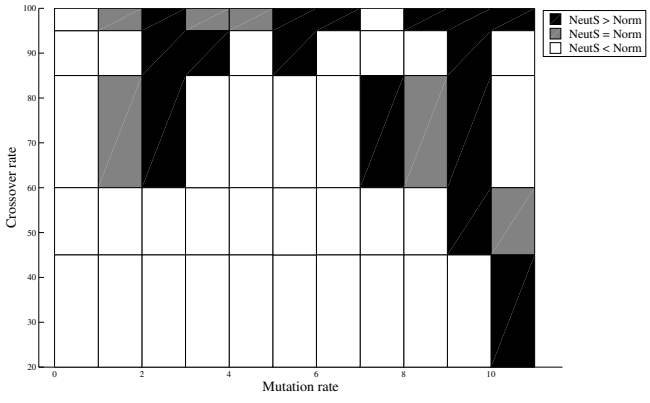

Figure 6: Best individual fitness comparison between normal runs and simple neutral ones. Squares are black where normal is better than simple neutral, white where simple neutral is better than normal and gray where the difference is statically insignificant.

When the fitness comparison is used, the advanced neutral series shows the best overall performance and it is better than the normal series in almost all the chose configurations. Only with very high crossover rates and high mutation rates the normal series is better than the neutral ones.

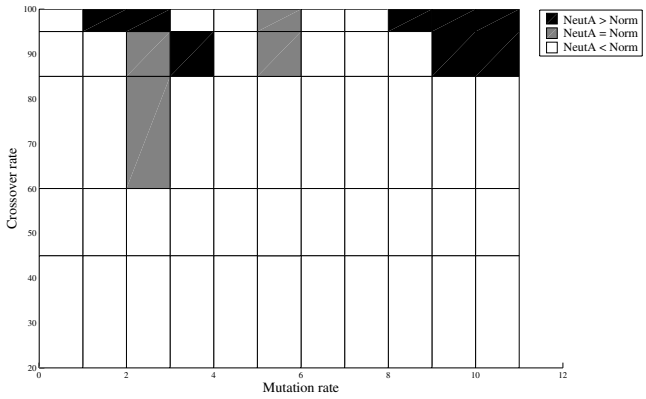

Figure 7: Best individual fitness comparison between normal runs and advanced neutral ones. Squares are black where normal is better than advanced neutral, white where advanced neutral is better than normal and gray where the difference is statically insignificant.

Figure 5 presents the plot of the average of the best fitness value against the mutation rate of the three series with a crossover rate of 60%.

This figure is significant because, for almost all the plotted mutation rates, the advanced neutral value is lower than the simple neutral and normal ones, only with a mutation rate of 5% the simple neutral slightly improves the advanced neutral value. The average of the difference between the normal and advanced neutral values with a crossover rate of 60% is 3.03.

Table 3 enumerates the smallest average of the best fitness value reached by all series. The difference between the smallest reached value by the normal and simple neutral series and the advanced neutral one is significant.

Table 3: Smallest average of the best fitness value reached.

Series	Crossover	Mutation	Fitness
Normal	100%	9%	2.82
Simple neutral	45%	7%	2.72
Advanced neutral	60%	7%	1.42

Table 4 lists effort values and cumulative success probability for the Santa Fe trail reported in the literature. Despite the effort value reached in this work is not very impressive, the cumulative success probability is remarkable. Only 8 runs out of 50 have not found an optimal solution, this value is improved only by Restricted EP [3], but it is significantly bigger to the other GP reported values.

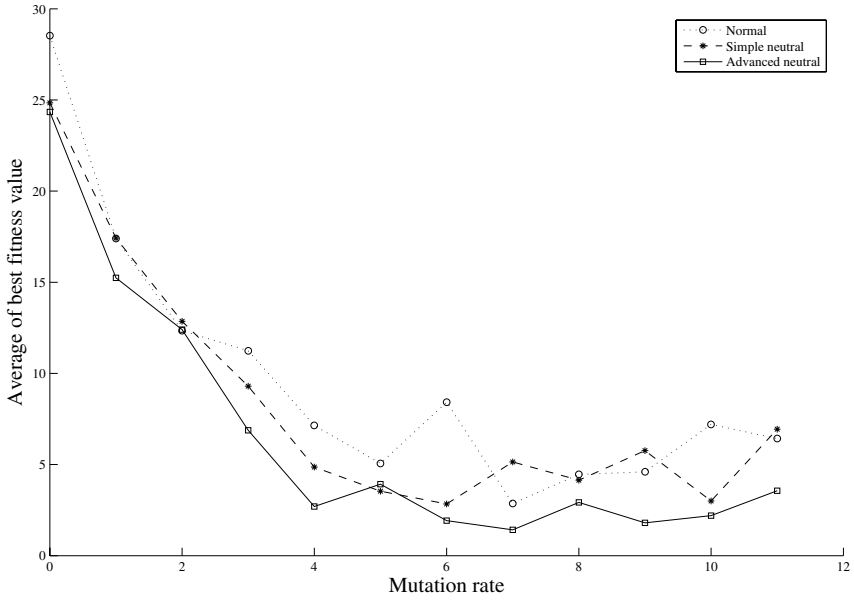

Figure 5: Average of the best fitness value vs. Mutation rate with a 60% crossover rate for normal, simple and advanced neutral series.

Table 4: Effort to solve Santa Fe trail.

Method	**I/1000**	**Cumulative success probability** $P(M,i)$
Koza GP [9]	450	48/111
PDGP [11]	336	-/-
Strict Hill Climbing 50%-150% mutation [11]	188	8/50
Neutral CGP [12]	173	32/100
Restricted EP [3]	136	47/50
Limiting to 1 food ahead [10]	120	19/50
Advanced Neutral, Crossover rate 60%, Mutation rate 5%	149	42/50

7. DISCUSSION

From the previous section it is clear that the effort do not significantly increase when neutral function is added. However the average of the best fitness value and the cumulative success probability show a clearly improvement when *Neut_progn* is included and more when the advanced mutation approach described in this paper is used too.

Figures 8 and 9 show two optimal individuals produced by the normal and neutral advanced series, respectively. It is important to remark the different size of the trees, this is caused by the chose GP depth parameters. But, this depth parameter selection was made concerning the different amount of inactive code of both representations.

The framed subtree of figure 9, is not active code in this individual (cause of the nearest *Neut_progn* chooser value) but it could be a building block of other optimal solutions (mutation of *Progn2*, see examples of [11]).

A drastic population behavior change could be performed if, at the GP process, occurs a crossover in the first edge or a referred chooser mutation. This situation is a cause of the better performance when the neutral function is added.

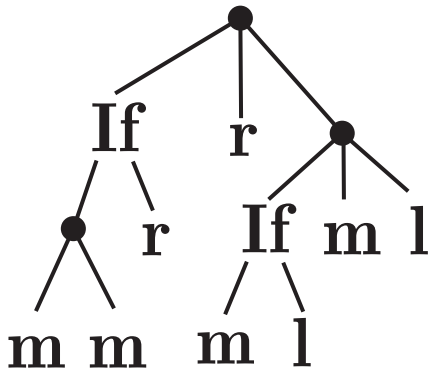

Figure 8: Example of optimal ant control program for the Santa Fe trail produced by the normal series

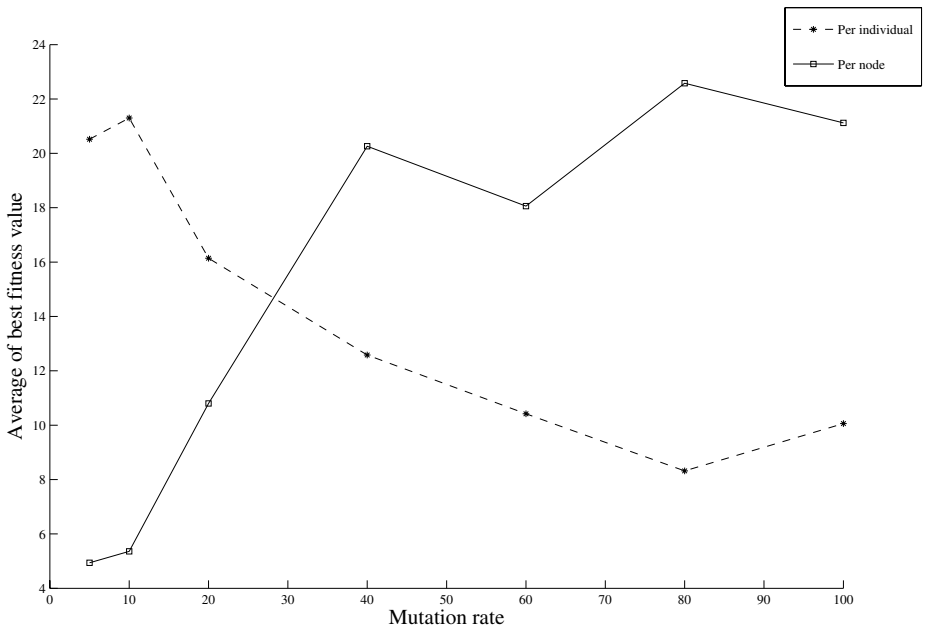

Figure 10: Average of the best fitness value vs. Mutation rate with a 60% crossover rate for per node point mutation and per individual subtree mutation.

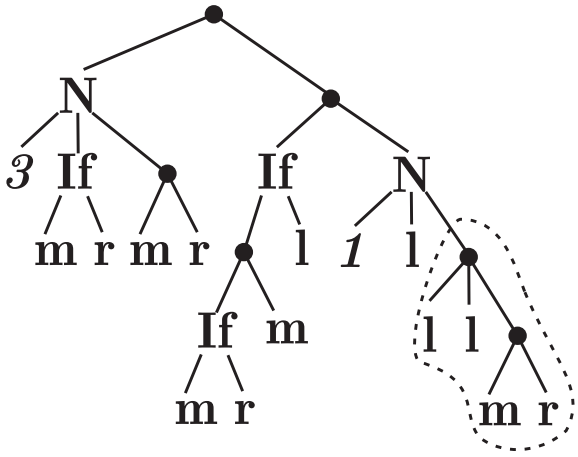

Figure 9: Example of optimal ant control program for the Santa Fe trail produced by the advanced neutral series

Finally, is important to differ per node mutation rate and per individual mutation rate. The first one is naturally related with point mutation and the second with subtree mutation, the cause of these relations is that each type of mutation is easiest implemented with the rate related to it.

For this problem and working with trees of depth 5, a 100% per individual mutation rate is equivalent to a value lower than 1.5% per node mutation rate.

To clarify this, an additional set of runs were performed with the parameters of table 2 and a 60% crossover rate, but with different percentages for per node point mutation and per individual subtree mutation. Figure 10 presents the plot of the average of the best fitness value against the mutation rate for the two mutation cases.

As expected, the average of the best fitness value decreases whit high per individual mutation (80%). On the contrary, this is the worst point for per node mutation.

8. CONCLUSIONS AND FUTURE WORK

Based on the use of the approach described in this paper, explicit neutrality has been allowed in the GP representation for the Santa Fe trail Artificial Ant Problem by the inclusion of a neutral function to the function set.

Several runs were performed in order to map the benchmark fitness landscape. Effort and average of the best fitness value comparisons were carried out.

Results empirically shown that despite the effort value reached by the advanced neutral method is not smaller to all the previous reported ones, the advantage over the original reported value is enough to suggest more work with the neutral function. Average fitness and cumulative success probability encourage this idea.

The difference between per node and per individual mutation rate was also empirically presented. More work must be done in order to understand the relation between both mutation cases.

Future work also includes the analysis of the combination of neutral function with other modifications to the Artificial Ant Problem and with variations to the evolutive process reported in the literature.

9. ACKNOWLEDGMENTS

The first author gratefully acknowledges the financial support provided by CONACYT (Consejo Nacional de Ciencia y Tecnología) and DGEP (Dirección General de Estudios de Posgrado), UNAM.

Authors also thanks PAPIIT, UNAM under the project IN115806-3.

10. REFERENCES

[1] W. Banzhaf. Genotype-phenotype mapping and neutral variation - a case study in genetic programming. In *Parallel Problem Solving from Nature III*, volume 866 of *Lecture Notes in Computer Science*, pages 322–332, October 1994.

[2] W. Banzhaf, P. Nordin, R. Keller, and F. Francone. *Genetic programming - An introduction*. Morgan Kaufmann, San Francisco, 1998.

[3] K. Chellapilla. Evolutionary programming with tree mutations: Evolving computer programs without crossover. In J. Koza, K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba, and R. L. Riolo, editors, *Genetic Programming 2007: Proceedings of the Second Annual Conference on Genetic Programming*, pages 432–438. MIT Press, Morgan Kaufmann, July 1997.

[4] M. Collins. Finding needles in haystacks is harder with neutrality. In H.-G. Beyer, U.-M. O'Reilly, D. V. Arnold, W. Banzhaf, D. Dasgupta, K. Deb, J. A. Foster, E. D. de Jong, H. Lipson, X. Llora, S. Mancoridis, A. M. Tyrrell, J.-P. Watson, and E. Zitzler, editors, *GECCO 2005: Proceedings of the 2005 conference on Genetic and Evolutionary Computation*, volume 2, pages 1613–1618. ACM Press, June 2005.

[5] E. Galván-López, K. Rodríguez-Vázquez, and R. Poli. Beneficial aspects of neutrality in GP. In F. Rothlauf, editor, *Late breaking paper at Genetic and Evolutionary Computation Conference (GECCO'2005)*, Washington, D.C., USA, June 2005.

[6] C. Janikow and C. Mann. CGP visits the santa fe trail-effects of heuristics on GP. In H.-G. Beyer, U.-M. O'Reilly, D. V. Arnold, W. Banzhaf, D. Dasgupta, K. Deb, J. A. Foster, E. D. de Jong, H. Lipson, X. Llora, S. Mancoridis, A. M. Tyrrell, J.-P. Watson, and E. Zitzler, editors, *GECCO 2005: Proceedings of the 2005 conference on Genetic and Evolutionary Computation*, volume 2, pages 1697–1704, Washington DC, USA, June 2005. ACM Press.

[7] D. Jefferson, R. Collins, C. Cooper, M. Dyer, M. Flowers, R. Korf, C. Taylor, and A. Wang. Evolution as a theme in artificial life: The genesys/tracker system. In C. G. Langton, C. E. Taylor, D. J. Farmer, and S. Rasmussen, editors, *Artificial Life II*, pages 549–578. Santa Fe Institute Studies in the Sciences of Complexity, Addison-Wesley, February 1992.

[8] M. Kimura. *The neutral theory of molecular evolution*. Cambridge University Press, 1983.

[9] J. Koza. *Genetic Programming: On the programming of computers by means of natural selection*. MIT Press, Cambridge, Massachusetts, 1992.

[10] W. Langdon and R. Poli. Better trained ants for genetic programming. Technical Report CSRP-98-12, University of Birmingham, School of Computer Science, April 1998.

[11] W. Langdon and R. Poli. Why ants are hard. In J. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D. B. Fogel, M. H. Garzon, D. E. Goldberg, H. Iba, and R. Riolo, editors, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 193–201, University of Wisconsin, Madison, Wisconsin, USA, July 1998.

[12] J. Miller and P. Thomson. Cartesian genetic programming. In R. Poli, W. Banzhaf, W. Langdon, J. Miller, P. Nordin, and T. C. Fogarty, editors, *Third European Conference on Genetic Programming*, volume 1802 of *Lecture Notes in Computer Science*, pages 121–132. Springer-Verlag, 2000.

[13] P. Smith and K. Harries. Code growth, explicitly defined introns and alternative selection schemes. *Evolutionary Computation*, 6(4):339–360, 1998.

[14] T. Soule. Operator choice and the evolution of robust solutions. In R. L. Riolo and B. Worzel, editors, *Genetic Programming Theory and Practice*, chapter 16, pages 257–270. Kluwer, July 2003.

[15] M. Tomassini, L. Vanneschi, F. Fernández, and G. Galeano. A study of diversity in multipopulation genetic programming. In P. Liardet, P. Collet, C. Fonlupt, E. Lutton, and M. Schoenauer, editors, *Evolution Artificielle, 6th International Conference*, volume 2936 of *Lecture Notes in Computer Science*, pages 243–255, Marseilles, France, October 2003. Springer-Verlag.

[16] T. Yu and J. Miller. Neutrality and the evolvability of boolean function landscape. In J. F. Miller, M. Tomassini, P. L. Lanzi, C. Ryan, A. G. B. Tettamanzi, and W. B. Langdon, editors, *Genetic Programming, Proceedings of EuroGP'2001*, volume 2038 of *Lecture Notes in Computer Science*, pages 204–217, Lake Como, Italy, April 2001. Springer-Verlag.

[17] T. Yu and J. Miller. Needles in haystacks are not hard to find with neutrality. In J. A. Foster, E. Lutton, J. F. Miller, C. Ryan, and A. Tettamanzi, editors, *Proceedings of the Fifth European Conference on GP*, volume 2278 of *Lecture Notes in Computer Science*, pages 13–25, Kinsale, Ireland, April 2002. Springer-Verlag.

[18] T. Yu and J. Miller. The role of neutral and adaptive mutation in an evolutionary search on the onemax problem. In E. Cantú-Paz, editor, *Late Breaking Papers at the Genetic and Evolutionary Computation Conference (GECCO-2002)*, pages 512–519, New York, USA, July 2002. AAAI.