Multi Niche Parallel GP with a Junk-code Migration Model

Santi Garcia[1], John Levine[2], and Fermin Gonzalez[3]

[1] Computer Science Department, University of Oviedo, SPAIN,
carbajal@lsi.uniovi.es
[2] University of Edinburgh. School of Informatics,
johnl@inf.ed.ac.uk
[3] INDRA. Alcala de Henares. Madrid. SPAIN,
fgmartinez@indra.es

Abstract. We describe in this paper a parallel implementation of Multi Niche Genetic Programming that we use to test the performance of a modified migration model. Evolutive introns is a technique developed to accelerate the convergence of GP in classification and symbolic regression problems. Here, we will copy into a differentiated subpopulation the individuals that due to the evolution process contain longer Evolutive Introns. Additionally, the multi island model is parallelised in order to speed up convergence. These results are also analysed.
Our results prove that the multi island model achieves faster convergence in the three different symbolic regression problems tested, and that the junk-coded subpopulation is not significantly worse than the others, which reinforces our belief in that the important thing is not only fitness but keeping good genetic diversity along all the evolution process. The overhead introduced in the process by the existence of various island, and the migration model is reduced using a multi-thread approach.

1 Introduction. Background

In this section we present the background knowledge about junk code growth, and multi niche Genetic Programming, that is the starting point to this study.

1.1 Junk Code

The benefits and disadvantages of existing non coding segments inserted in the genotype of the individuals of a population have been a theme of discussion in the GP community during the last years. Some of these studies show that introns can grow up excessively and exhaust the memory of the system, [1]. In other studies, introns seem to be useful as they appear as the evolution process goes on, protecting good building blocks from the destructive effect of the crossover operator [2], [3], [4], [5], [6]. In these studies, introns seem to be non random code which, given an appropriate crossover, can increase the fitness of the coding sections.

In [7], a strategy to promote the automatic growth and shrinking of non coding segments in the individuals of a GP population is proposed. The method consists of adapting the probability of selection of every node in the tree after each application of the crossover operator. Artificially generated introns are called Evolutive Introns since as the process goes on they usually grow, and shrink in different phases of the learning process.

In [8], the three hypothesised causes of code growth are examined. Removal Bias and Protective hypotheses come reinforced after this study, while drift hypothesis is not supported by this experiments.

1.2 Multi Niche Genetic Programming

The use of multiple populations, or niches, is called *distributed genetic algorithm*, or *island model*. In [9], two different Multi niche Genetic Programming techniques are tested. In the first one, (MP ring architecture), each subpopulation chooses its five best individuals and sends it to the next subpopulation in the ring. Each subpopulation takes the individuals sent to it, and uses them to replace its five worst members. The second, (injection architecture) is a hierarchical arrangement. For example, in a 3 subpopulation model, two subpopulations would send their best five individuals to the third one, that would use this ten individuals to replace its worst ten [10]. The most important thing when setting up a multi niche system is the migration model, and its parameters, as stated in [11].

2 Multi Niche Parallel GP

In[12], D. Andre and John Koza studied the effects of parallelisation on GP comparing the results obtained using different migration rates with the ones obtained using only one panmictic population. The study of these performance variations is absolutely out of the scope of this work.

We will study the behaviour of a multi island model of four different islands (POP0, POP1, POP2, and POP3), and a panmictic isolated population of size POP1 + POP2 + POP3.

The system works as follows: upon completion of a pre-defined number of generations (a parameter of the system), a certain number of individuals in POP1, POP2 and POP3 will be probabilistically selected for migration to each one of the other two subpopulations (see figure 1). Each one of these populations will contribute to the migration process with the same number of individuals. The amount of migration will be another parameter of the system. Each time this migration process occurs, the same number of individuals will be selected from the three island model, and copied into the population named POP0, using the value of the longest intron contained in each individual as selection criterion. We will not make an exhaustive study of the results obtained with different migration rates, because of the extension of the paper, and we will focus our study in the behaviour of the panmictic model, the enriched population, and the classic 3 island model. From now on, we will refer to these as Panmictic, POP0, POP1, POP2,and POP3 subpopulations, respectively, where POP0 is the one where we will copy the individuals with longer introns.

Our Hypothesis is that POP0 will converge as well as PO1, POP2, POP3 because the individuals with high value introns represent useful genetic material in the population.

2.1 Parallelisation Strategy

We used a two-level parallelisation approach. With the help of standard POSIX threads we can create a separate thread of execution for each population. For the parallelisation of the process of passing from one generation the the next, inside each population we tried to use MPI library [13], [14] but mixed code using POSIX threads and MPI was not reliable at all, so we decided to keep the multithtread POSIX based model, and use MPI only for parallelising the repetition of experiments. The main process is responsible for creating a process for each subpopulation (three subpopulations, the one breeded with higher value introns, and the panmictic one), and synchronise the evolution of each one, directing the migration process when the adequate number of generations have passed. Each one of these secondary processes is responsible for carrying on the evolution on each subpopulation, (see figure 1). In this work we will use the migration model proposed by J. Koza and D. Andre, and will introduce one more population where we will copy the individuals with larger Evolutive Introns [7]. We will carry on evolution simultaneously in a panmictic population whose size will be three times the niches. This way, we will try to analyse the behaviour of an Evolutive Intron based migration model.

2.2 Experimental Setup

In this section we describe the three problems we chose to test the performance of the multi niche model.

Rosenbrock's function Rosenbrock's valley [2], also known as Banana function, is a classic optimisation problem. The global optimum is inside along, narrow, parabolic shaped flat valley(see figure 2). To find the valley is trivial, however convergence to the global optimum is difficult and hence this problem has been repeatedly used in assess the performance of optimisation algorithms.

$$f(x,y) = (x-1)^2 + 10(x^2 - y^2) \tag{1}$$

We will take the value of the function in 2500 points along the XY plane between -2.4 and 2.4, to compute the value of the fitness function.

Gudermann's function The Gudermannian function gd(x) is a relative of the circular and hyperbolic functions, namely, the inverse tangent of sinh(x). It is shown that the motion of a simple pendulum released from its equilibrium position with just enough kinetic energy to reach the inverted position is naturally

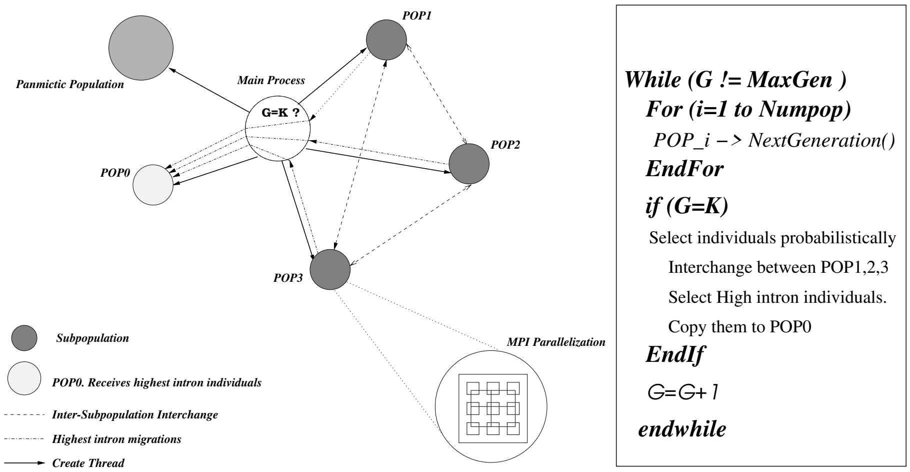

Fig. 1. Two-level parallelisation. The main process is responsible of the creation and synchronisation of the on going different evolution processes.

Table 1. Algorithm parameters. Rosenbrock Function.

Objective	Symb. Reg $Ros(x,y) = (x-1)^2 + 10(x^2 - y^2)$
Function set	+, -, *, DIV, SIN, COS
Terminal set	X, Y
Fitness cases	2500 samples in[-2.4,2.4]
Parameters	M=1000, MPOP0,MPOP1,MPOP2,MPOP3 = M/3
Error	$\sum_{i=1}^{2500} \lvert Ros(x_i,y_i) - f(x_i,y_i) \rvert$

described by this function. This non-periodic solution of the pendulum equation can be viewed as completing the family of periodic solutions classically expressed by elliptic functions. See figure 2.

$$f(x) = \ln \frac{1}{\cos x + \tan x} \tag{2}$$

Table 2. Algorithm parameters. Gudermann function.

Objective	Symb. Reg $Gud(x) =$
Function set	+, -, *, DIV, SIN, COS
Terminal set	X
Fitness cases	1000 samples in[-1.5,1.5]
Parameters	M=1000, MPOP0,MPOP1,MPOP2,MPOP3 = M/3
Error	$\sum_{i=1}^{1000} \lvert Gud(x_i) - f(x_i)\rvert$

Koza's symbolic regression problem The third problem is the symbolic regression on

$$Koz(x) = x^4 + x^3 + x^2 + x \tag{3}$$

as stated [15].

Table 3. Algorithm parameters. Koza's symbolic Regression Problem.

Objective	Symb. Reg $Koz(x) = x^4 + x^3 + x^2 + x$
Function set	+, -, *, DIV,SIN, COS
Terminal set	X
Fitness cases	20 samples in[-1,+1]
Parameters	M=500, MPOP0,MPOP1,MPOP2,MPOP3 = M/3
Error	$\sum_{i=1}^{20} \lvert Koz(x_i) - f(x_i)\rvert$

2.3 Results

This section summarises the results obtained when applying this multi niche approach on the problems described in section 2.2

Figure 3 (left) shows the medium error in panmictic populations were size M= 1000, and 1000/3 for POP0, POP1, POP2,and POP3, when applying the

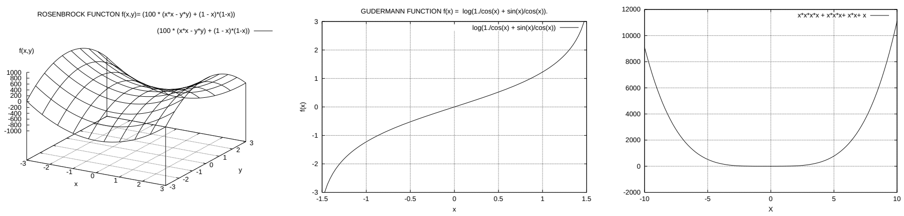

Fig. 2. Rosenbrock function(left). Gudermann function (center). Koza's symbolic regression problem (right).

approach to Rosenbrock's function. It is clear that, in this problem, the panmictic approach was clearly inferior to the multi niche island model. The same figure (right) shows the averaged percentage of individuals that would have been selected for migration using both methods (standar fitness, or Evolutive Introns length). This study was made in order to determine if the population named POP0 was really receiving different migrated individuals than the selected if the migration criterion was only fitness. Figures 4(center) and 5(center) show the same behaviour in the other two benchmark problems. The percentage of coincidence between fitness and introns-length criteria is rarely over 40%.

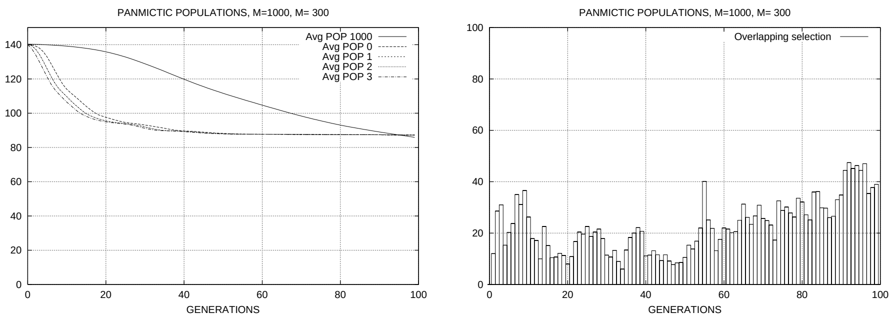

Fig. 3. Rosenbrock function. Results.

Figures 4 and 5 show coherent results when using Gudermann function and Koza's symbolic regression problem.

Figure 6 shows the important difference between the average cumulative cpu time needed to reach generation 100 in panmictic, multi island threaded, and non threaded model. This is due to the overload generated by the process of looking for the longest intron individuals. The difference between panmictic and multipopulation models is stretched when threads are enabled.

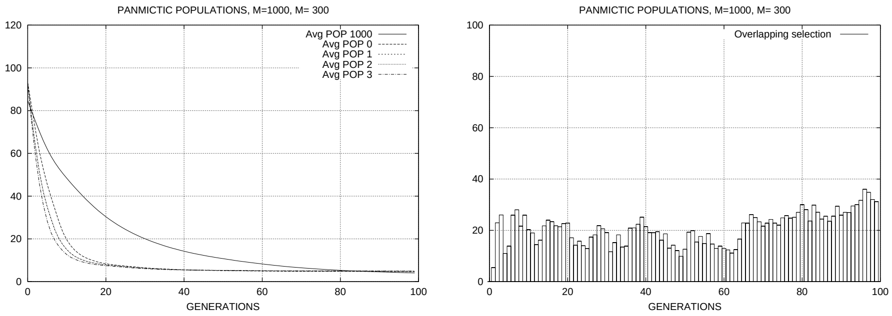

Fig. 4. Gudermann function. Results.

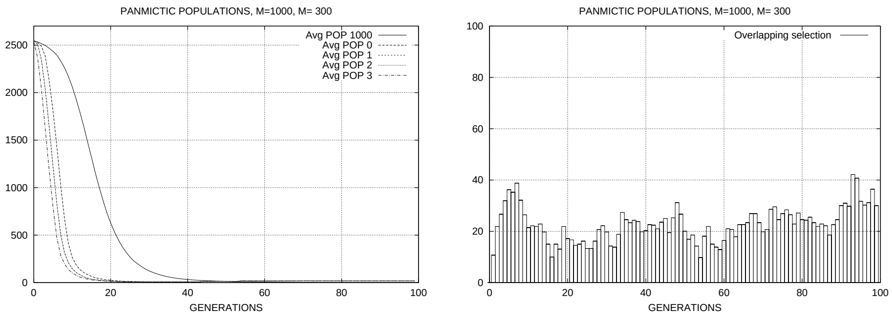

Fig. 5. Koza's symbolic regression problem. Results.

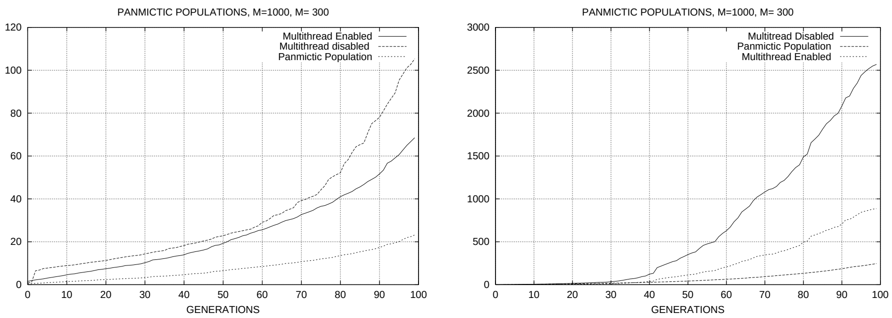

Fig. 6. Accumulated cpu time needed to reach generation 100.

Finally, as the experiments on Rosenbrock's function with only 100 generations did not show clearly if the evolution was finished or not (see figure 3), we repeated this runs over 200 generations with the results shown in figure 7. In this figure, it can be seen that panmictic population is still decreasing medium error in generation 200. Multi niche model is stuck after a very fast initial error decreasing. This is probably due to a insufficient population size of the islands.

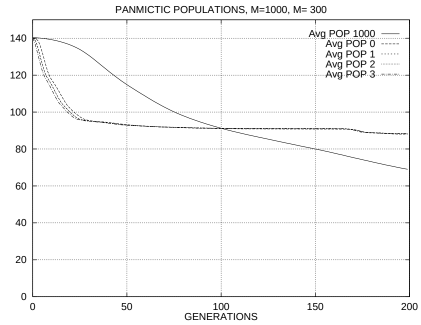

Fig. 7. Rosenbrock function. Results over 200 generations. Panmictic population reaches better results due to a bad selection of the sub populations size.

3 Conclusions

We have obtained coherent results in the three selected symbolic regression experiments. Based on our results, our conclusions can be summarised in

1. As we stated in our hypothesis, a subpopulation breeded with selected individuals according to a different selection criterion, as intron length, can converge as fast the ones in which the selection criterion was fitness function.
2. Multi Niche Genetic Programming can improve the convergence in Symbolic Regression problems, even if sometimes some kind of genetic drift could appear. This problem is probably due to a bad selection of population size.
3. The use of a multithreaded model reduces time performance degradation caused by the extra processing work needed to carry on evolution in a multi-niche system.

4 Acknowledgements

The set of experiments described in this work were conducted using the machines of the EPCC, in the University of Edinburgh. Mainly, the runs were executed

in a Sun HPC cluster. It comprises an eight processor 400 MHz UltraSPARC-II system , with 8 Gbyte of shared memory and 276 Gbyte of disc space. This machine acts as front-end to three Sunfire 6800 systems two of which have twenty four 750 MHz UltraSPARC-III processors and 48 Gbyte of shared memory, and the third of which has sixteen such processors.

The authors would like to acknowledge the support of the European Community Access to Research Infrastructure action of the Improving Human Potential Programme (contract No HPRI-CT-1999-00026), and the people from EPCC at Edinburgh University.

References

1. Andre, D., Teller, A.: A study in program response and the negative effects of introns in genetic programming. In Koza, J.R., Goldberg, D.E., Fogel, D.B., Riolo, R.L., eds.: Genetic Programming 1996: Proceedings of the First Annual Conference, Stanford University, CA, USA, MIT Press (1996) 12–20
2. Wineberg, M., Oppacher, F.: The benefits of computing with introns. In Koza, J.R., Goldberg, D.E., Fogel, D.B., Riolo, R.L., eds.: Genetic Programming 1996: Proceedings of the First Annual Conference, Stanford University, CA, USA, MIT Press (1996) 410–415
3. Levenick., J.: Inserting introns improves genetic algorithm success rate: Taking a cue from biology. Proceedings of the Fourth International Conference on Genetic algorithms **1** (1991) 123–127
4. Haynes, T.: Duplication of coding segments in genetic programming. In: Proceedings of the Thirteenth National Conference on Artificial Intelligence, Portland, OR (1996) 344–349
5. P.Nordin, Francone, F., Banzhaf, W.: Explicitly defined introns and destructive crossover in genetic programming. In Rosca, J.P., ed.: Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications, Tahoe City, California, USA (1995) 6–22
6. Angeline, P.J.: Genetic programming and emergent intelligence. In Kinnear, Jr., K.E., ed.: Advances in Genetic Programming. MIT Press (1994) 75–98
7. Garca, S., Gonzlez, F.: Evolutive introns: A non-costly method of using introns in GP. Genetic Programming and Evolvable Machines **2** (2001) 111–122
8. Soule, T., Heckendorn, R.B.: An analysis of the causes of code growth in genetic programming. Genetic Programming and Evolvable Machines **3** (2002) 283–309
9. Kirkwood, I.M.A., Shami, S.H., Sinclair, M.C.: Discovering simple fault-tolerant routing rules using genetic programming. In: ICANNGA97, University of East Anglia, Norwich, UK (1997)
10. Punch, B.: Royal trees as a benchmark problem for genetic programming: A parallel processing example. In Angeline, P.J., Kinnear, Jr., K.E., eds.: Advances in Genetic Programming 2. MIT Press, Cambridge, MA, USA (1996) –
11. Fernandez, F., Tomassini, M., Punch III, W.F., Sanchez, J.M.: Experimental study of multipopulation parallel genetic programming. In Poli, R., Banzhaf, W., Langdon, W.B., Miller, J.F., Nordin, P., Fogarty, T.C., eds.: Genetic Programming, Proceedings of EuroGP'2000. Volume 1802 of LNCS., Edinburgh, Springer-Verlag (2000) 283–293

12. Andre, D., Koza, J.: A parallel implementation of genetic programming that achieves super-linear performance. In: Proceedings of int. Conf. on Parallel and Distributed Processing Techniques and Applications, In Arabnia, Hamid R. (editor) (1996) –
13. Forum, M.P.I.: MPI: A message-passing interface standard. Technical Report UT-CS-94-230 (1994)
14. MPIF, M.P.I.F.: MPI-2: Extensions to the Message-Passing Interface. Technical Report, University of Tennessee, Knoxville (1996)
15. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, MA, USA (1992)