Dynamic Populations and Length Evolution: Key Factors for Analyzing Fault Tolerance on Parallel Genetic Programming

Daniel Lombraña Gonzalez
University of Extremadura
Merida, Badajoz, SPAIN
daniellg@unex.es

Francisco Fernandez de Vega
University of Extremadura
Merida, Badajoz, SPAIN
fcofdez@unex.es

ABSTRACT

This paper presents an experimental research on the size of individuals when fixed and dynamic size populations are employed with Genetic Programming (GP). We propose an improvement to the Plague operator (PO), that we have called Random Plague (RPO). Then by further studies based on the RPO results we analyzed the Fault Tolerance on Parallel Genetic Programming.

Categories and Subject Descriptors

Track [**Genetic Programming**]; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods and Search—*heuristic methods*

General Terms

Management, Measurement, Experimentation, Performance, Reliability

Keywords

Genetic Programming, Size evolution, Parallel and Distributed EAs

1. OVERVIEW OF THE WORK

The growth of individuals in GP is considered one of the main drawbacks when applying the technique to optimization problems. For reducing the individuals' growth we can use the PO. This operator deletes the worst individuals from the population (worst fitness values). We wanted to know the real impact of this operator at an individual level and with this purpose we set up several experiments. The results showed that the worst individuals have smaller sizes than better individuals. As a consequence the PO is eliminating the smallest individuals that also have lower probabilities of being selected for generating offspring. So if we want to reduce as much as possible the size of the best individual we should change the match criteria of the PO. Thus we propose an improvement for the PO: the Random Plague Operator. The *modus operandi* is to select randomly an individual for deleting it. In the first series of experiments the RPO produced, for the same computational effort, equal or better quality solutions than classic PO.

GECCO'07, July 7–11, 2007, London, England, United Kingdom.
ACM 978-1-59593-697-4/07/0007.

Table 1: Fitness/Effort for a Population size of 1000.

(a) Even Parity 5. 2% Elimination

	Effort $3.75 * 10^6$		
	Classic	Plague	Random
Best Ind. Fitness	9.72	9.44	9.28
Population Average Length	140.32	169.96	192
Generation	56	68	99

(b) Symbolic Linear Regression. Elimination of 10%

	Effort $84 * 10^3$		
	Classic	Plague	Random
Best Ind. Fitness	0.26	0.24	0.19
Population Average Length	8.95	9.30	12.45
Generation	11	12	54

Those results leaded us to go a step further and study the results from a totally different point of view: Parallel Genetic Programming. If we are planning to use a parallel version of our algorithm we will have to deal with resources failures (microprocessor errors, hangs, overflows, etc.). In order to circumvent these problems people are usually employing an error library for handling them. Nevertheless given that the RPO randomly deletes individuals from the population, we can interpret those removed individuals as the resource failures. Each removed individual is a lost/error computer in a simulation of a Fine Grained Parallel GP. This suggests us that GP could be Fault-Tolerant by nature. With this new approach we executed several experiments with different failure percentages for comparing it with the free-failure environment. The results showed (Table 1) that even though we are loosing resources we are still getting good enough solutions, at least the same quality and sometimes better results, as compared with the non-error environment [1], and most importantly without using any kind of error handling library; we are simply ignoring the failures. In conclusion we have presented an improvement to the PO called RPO. Based on the RPO results we have done other experiments that suggests to us that GP is Fault-Tolerant by nature on a parallel and distributed environment.

2. REFERENCES

[1] D. Lombraña Glez. and F. Fernández de Vega. On the intrinsic fault-tolerance nature of parallel genetic programming. *The 15th. Euromicro Conference on Parallel, Distributed and Network-based Processing*, 2007. pp. 450-458. IEEE CS.