Compression of Effective Size in Genetic Programming

Peter Nordin
Physical Resource Theory
Chalmers University of Technology
S-41296 Göteborg, Sweden
nordin@fy.chalmers.se

Wolfgang Banzhaf
LS11
Univeristy of Dortmund
D-44221 Dortmund, Germany
banzhaf@icd.de

Frank D. Francone
RML Inc.
Grand Ave., Oakland, USA
frank@francone.com

Abstract

We *measure* the compression of information in a genetic programming system. The investigation is performed taking *introns* in the genome into account. We mainly investigate evolution of linear computer code but also present results from evolution of tree structures as well as messy genetic algorithms. The size of solutions is an important property of any system trying to learn or adapt to its environment. Our results show compression or constant size of exons during evolution—in contrast to the rapid growth of overall size. Our conclusion is that an *built-in pressure* towards low-complexity solutions is *measurable* in several simulated evolutionary systems which may account for the robust adaptation showed by these systems.

1 Introduction

DNA works principally by coding for the production of proteins, polypeptides or RNA. One of the many curiosities of biology is that *over 70 %* of the DNA base pairs in eucaryotic life forms do not produce proteins, polypeptides or RNA, [8]. While some of these DNA base pairs are undoubtedly control sequences that turn other genes on and off, most of them appear to do nothing at all. Biologists refer to much of this code as *'introns'* or *'junk DNA'*. The fascinating fact is that introns emerge in genetic programming as well. The evolution of such code fragments has been repeatedly observed by GP researchers in tree–based GP–some refer to the phenomenon as *bloat* , [7], [1]. Some researchers argue that introns appear in GP populations because they have an important function in evolution while others argue that they are a useless side–effect of evolution [4].

Evolution in nature as well as in computer simulation is a form of learning. Compression of information is an important concept in the theory of learning. We argue for the hypothesis that there is an inherent compression pressure towards short, elegant and general solutions in variable length evolutionary algorithms and potentially in natural evolution. This pressure becomes visible if the size or complexity of solutions is measured without introns. The built in "parsimony" selection pressure influences *generality* and *adaptiveness* of the phenotype. Some of these effects are positive and some are negative.

The positive effect of compact information representation is often referred to as Occam's Razor. The principle of Occam's Razor, formulated 700 years ago, states that from two possible solutions to a problem we should choose the shorter one. We would like to argue that one of the foundations of evolution in general and Genetic Programming in particular is that they have the built in property of favoring short solutions and sub-solutions. This property may be one of the reasons that evolution works so efficiently and robustly in a diverse set of domains. The compression property could also be responsible for the ability of a solution to be generic and applicable to a larger set of data than the set of seen training data (selection events).

Here we also try to *measure* the compression of the active part of the genome. We have used a variant of GP operating on a string of bits to evolve an algorithm or program for a register machine [5]. The structures that undergo evolution are variable length strings of 32 bit instructions for a register machine (CPU). Genetic programs do not seem to favor parsimony in the sense that the evolved program structures become short and elegant measured with the absolute size of an individual. The crucial point is to measure the size of the code that are exons (not introns) called *effective length* instead of *absolute length.* Observing the effective length indicates that the evolutionary system not only favors parsimonious solutions for the final result, but constantly, for sub-solutions during the evolution of the population.

2 Problem set-up

We decided to study the compression hypothesis on three standard problems. The problem domains are collected from the machine learning literature [2]. Two of these are real world problems while one is a difficult artificial data set. The classification problems were cast into symbolic regression problems. Each class was given a range. We performed an extensive batch of test runs consisting of 420 complete AIM-GP runs with a population of 3000 each, evaluating several hundred billion test cases.

- Phoneme Recognition Data Set:The phone recognition problem is the same problem used for comparison between a register machine and a tree–based GP system mentioned in the previous chapter. The Phoneme recognition database contains two classes of vowels from isolated syllables spoken by different speakers. Each input training vector is five dimensional and describes different frequency properties of a vowel.

- The Gaussian 3D Data Set: Gaussian 3D is an artificially generated Classification problem. It has a three dimensional input and two different output classes. Class 0 is a set of points with a normal distribution across the three input axes with zero mean and standard deviation of 1. Class 1 is a similar series of points except that the standard deviation is 2 [2].

- The Iris Data Set: The IRIS data set is a well know classification benchmark from the machine learning community. The data set contains 150 examples of properties of iris plants which shall be classified into three different plant types.

3 Empirical Results

Average fitness during evolution of a symbolic regression GP run, is plotted in Figures 1 while 2, shows the evolution of absolute length and effective length in the same experiment.

We measure introns by changing each instruction in the genome to a NoOperation (NOP) and if this does not make any difference for *any* of the fitness cases then we call it an intron. For the tree-based-GP we look at values coming in to each node and the values

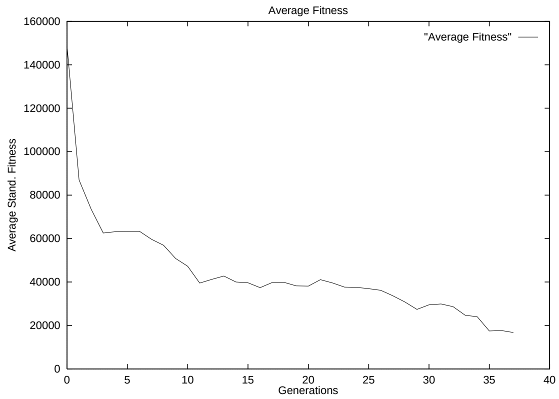

Figure 1: Average fitness

that propagates upwards from each node. If one value coming to the node is the same as the value out from the node *for all fitness cases* then the other sub-trees under the node are introns.

All runs observed end with exponential intron growth, if given enough time(30-800 generation) to proceed.

We tried the compression hypothesis on the 480 runs and 3 problems. Compression was measured from the point in evolution where the fitness value stabilized and the evolution of the effective size from this point was taken as the compression measure. 64 runs ended with increased effective size (most of them a few percent), 100 runs decreased, 316 runs remained constant. Most runs where effective size increased show some problem with early termination or bad fitness. A majority of runs showed stable effective length or *compression*. Runs which increased terminated 9 generations earlier on average than the other runs. The constant or decreasing runs usually displayed this behaviour for several tens of generations before the run was terminated by an exponential growth in absolute size (bloat). The average compression was 4%.

3.1 Other Evolutionary Techniques

The results may be extended to a more canonical GP system with hierarchical crossover and tree representation. We have so far done initial experiments with a tree–based GP system doing symbolic regression. The results indicate a distribution of destructive crossover similar to that in the *linear representation GP system* distribution. Figure 4 shows the distribution of crossover effect of tree based GP system doing symbolic regression of a polynomial over 60 generations. Figure 3 shows the evolution of absolute and effective length in the same experiments. Few issues in our assumption are specific to *linear representation GP sys-*

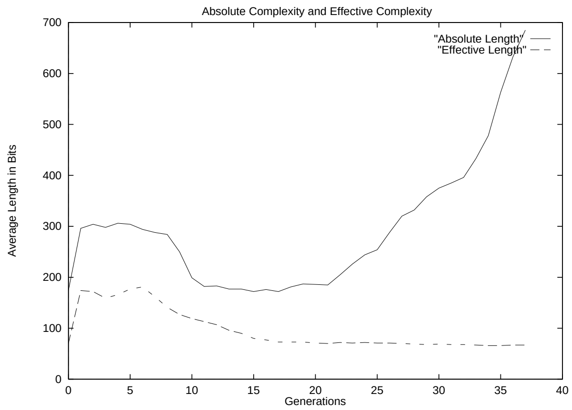

Figure 2: Average size

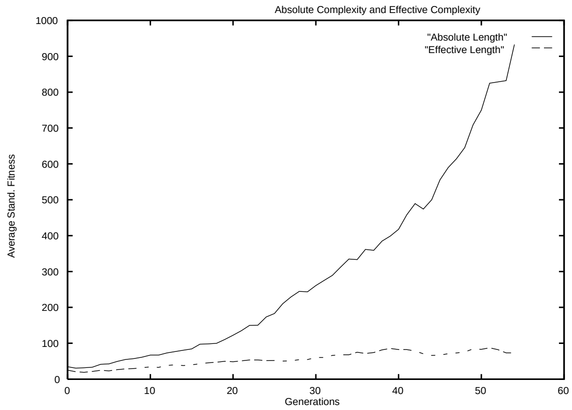

Figure 3: Tree–based GP

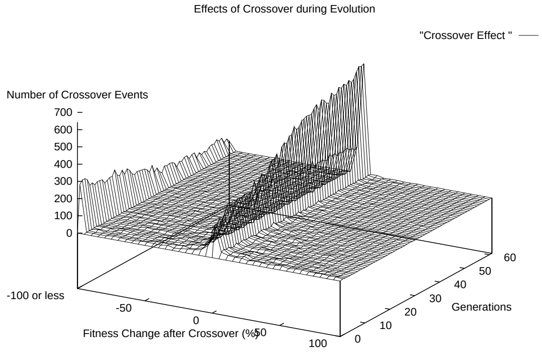

Figure 4: Crossover effects in S-Expression style GP

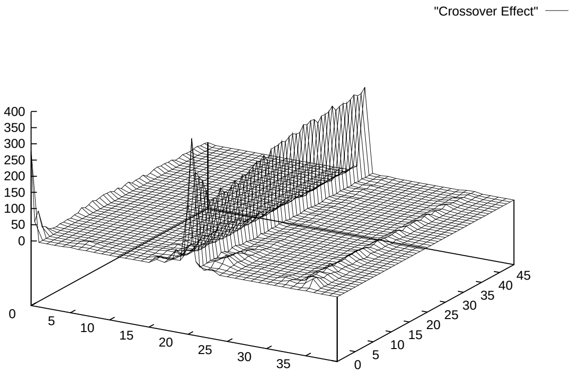

Figure 5: Messy-GA

tem which together with results such as those in Figure 4 suggests that our results may apply to a wider domain of systems, see also [6]. Another evolutionary algorithm with varying length binary genome is the *messy genetic algorithms* [3]. Initial experiments with this representation method for optimisation problems show similar results and a rapid growth of introns, see Figure 5.

4 Conclusions

In this paper we have measured *compression* in simulated evolution of structures. We have investigated the compression factor over a very large number of simulations (480) and three different domains. We have also showed similar compression behavior for other types of simulated evolution—messy-genetic algorithms and tree-structures. All results show a similar pattern where the absolute length of an individual grows rapidely while the *used* part of the genome stays constant (after some time), decreases or grows only very slowly. A possible explanation of this phenomenon is the need to protect the genome from the destructive effects of the genetic operators. The advantage of this compression phenomenon is that shorter solutions in often generalize better this could be one explanation of the power of evolutionary search, possibly extendable *beyond simulated evolution*—to natural evolution.

Acknowledgement

Peter Nordin gratefully acknowledges support from the Swedish Research Council for Engineering Sciences, NUTEK and the Swedish Institute.

References

[1] Angeline P.J. (1994) Genetic Programming and Emergent Intelligence In *Advances in Genetic Programming*, K. Kinnear, Jr. (ed.), Cambridge, MA: MIT Press.

[2] ELENA partners, Jutten, C., Project Coordinator 1995.Esprit Basic Research Project Number 6891, Document Number R3-B1-P.

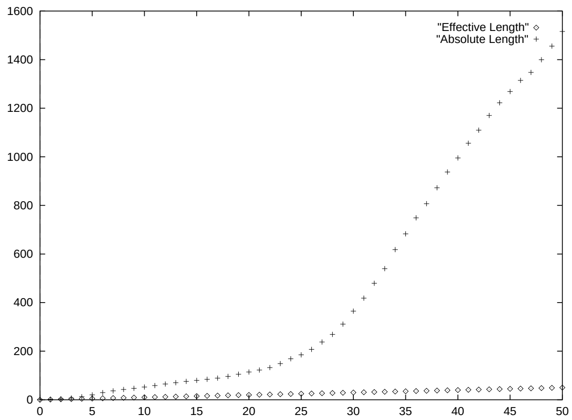

Figure 6: Messy-GA effective and absolute length

[3] Goldberg D E, Korb B and Deb K (1989) Messy genetic algorithms: Motivation analysis and first results *Complex Systems* **3** pp 493-530.

[4] Langdon, W.B., Soule, T., Poli, R. and Foster, J.A. (1999) The evolution of size and shape. In Spector, Langdon, O'Really and Angeline, editors, Advances in genetic Programming 3, chapter 8, pages 163-190. MIT Press, Cambridge, MA, USA,May 1999, Forthcomming

[5] Nordin .J.P. (1994) A Compiling Genetic Programming System that Directly Manipulates the Machine-Code, in: *Advances in Genetic Programming*, K. Kinnear, Jr.(ed.), MIT Press, Cambridge, MA

[6] Rosca J.P. (1995) Entropy-Driven Adaptive Representation. In *Proceeding of the GP workshop at Machine Learning 95, Tahoe City, CA*, J. Rosca (ed.),University of Rochester Technical Report 95.2, Rochester, NY, p. 23-32.

[7] Tackett W. (1994) Recombination Selection and the Genetic Construction of Computer Programs. *Dissertation: Department of Electrical Engineering Systems*, University of Southern California, Los Angeles, CA.

[8] Watson J.D, Hopkins, N.H, Roberts J.W, Wiener A.M, (1987) *Molecular Biology of the Gene*, Menlo Park, CA: The Benjamin/Cummings Publishing Company, Inc.