Variable Length Genetic Algorithms with Multiple Chromosomes on a Variant of the Onemax Problem

Investigating Changes in Chromosome Length

Rachel Cavill
University of York
York,UK
rc145@ohm.york.ac.uk

Stephen L Smith
University of York
York,UK
sls5@ohm.york.ac.uk

Andy M Tyrrell
University of York
York,UK
amt@ohm.york.ac.uk

ABSTRACT

The dynamics of variable length representations in evolutionary computation have been shown to be complex and different from those seen in standard fixed length genetic algorithms. This paper explores a simple variable length genetic algorithm with multiple chromosomes and its underlying dynamics when used for the onemax problem. The changes in length of the chromosomes are especially observed and explanations for these fluctuations are sought.

Categories and Subject Descriptors: G.1.6 Numerical Analysis: Global Optimization I.2.8 Artificial Intelligence-Control methods and search

General Terms: Algorithms Performance Design

Keywords: Genetic Algorithms, representations, size

1. INTRODUCTION

The one max problem is often used to explore the basic dynamics of operators within evolutionary computing and how they affect the formation of building blocks. This paper uses a variation of the problem to examine a variable length genetic algorithm under varying conditions to inform us about the underlying dynamics of the system and the settings which allow optimal behaviour.

The system used for the experiments in this paper was originally designed to be a *multi-chromosomal genetic programming* system [1, 2], where variable length strings are evolved and then *translated* into a Prolog program using a grammatical evolution [6] type process.

1.1 The Problem and Setup

The onemax problem, is a common test used throughout evolutionary computation. It is chosen for its simplicity and its ability to highlight whether an algorithm can optimise many variables in parallel. The standard one max problem is formulated on a fixed length binary string and is defined as $\sum_{i=0}^{n} x_i$ where n is the length of the binary string.

For a variable length genetic algorithm the onemax problem can be sensibly redefined as $1 - \frac{\sum_{i=0}^{n} x_i}{n}$ where n is the length of the binary string (low fitness is good).

There have been several other investigations into the onemax problem using variable length representations [3, 5, 7].

GECCO'06, July, 8–12, 2006, Seattle, WA, USA.
ACM 1-59593-186-4/06/0007.

The system used in these experiments involves individuals made up of 4 sets of 4 variable length binary strings. The operators used are *shuffling*, *crossover* and *mutation*. Selection is done via tournament selection with a tournament size of 4. The population has 100 individuals and all chromosomes start with at least 120 bits.

Shuffling involves taking the chromosomes from both parent individuals and within sets of 4, mixing them up to create two new sets in which every chromosome from the parents is present in its exact same form.

Crossover is like the standard GA crossover, except it is done between two strings from the same set and can be multi-point (up to 3 points are currently allowed within the system). The position of the crossover point is the same in both strings when counting from the left hand side of the string. Where not otherwise stated the crossover rate is 0.5.

Mutation can effect one of three actions, delete the digit at the locus to be mutated ($\frac{1}{4}$), flip the digit at the locus to be mutated ($\frac{1}{2}$) or insert a new digit at the locus to be mutated ($\frac{1}{4}$). Mutation normally acts on 1 in 500 bits.

2. RESULTS

In figure 1 we see that mutation is necessary; without it evolution is stunted after several generations and does not proceed. We observe two phases to the evolutionary process with mutation being irrelevant in the first phase (about 10-20 generations), but having a greater effect later in the run.

As shown in figure 2, a high crossover rate is beneficial to this problem and the correlation between performance and crossover rate is very strong. The headless chicken crossover operator shown in figure 1 is useless, which shows that crossover is doing much more than just macromutation.

The length of the chromosomes being evolved changes in interesting ways throughout the run (figure 3) and again shows evidence of two phases. The first phase lasting 20-30 generations and consisting of a sharp decline in length, before a drastic slowing of this decline occurs and the change in length stabilises. As shown, the mutation operator is not causing the shrinkage as, although it occurs slightly less when mutation is turned off, it is still evident. This lessening effect may also be triggered by switching to one mutation per chromosome instead of one mutation per 500 bits.

These results relate to a theory seen in the work done by Harvey [4] which predicts that when operators allow for jumps in length then "an early population could fluctuate in length through *long jump* adaptation which effectively acts

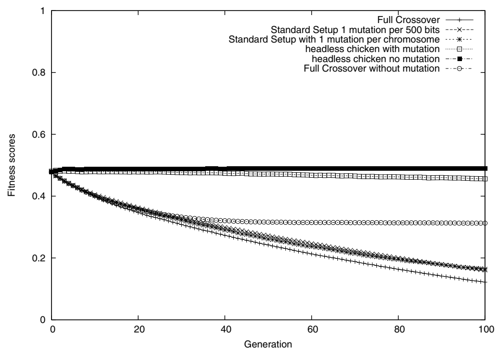

Figure 1: The effect of mutation within the system

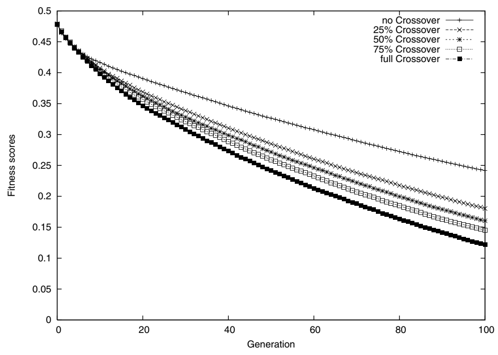

Figure 2: Increasing crossover rate helps

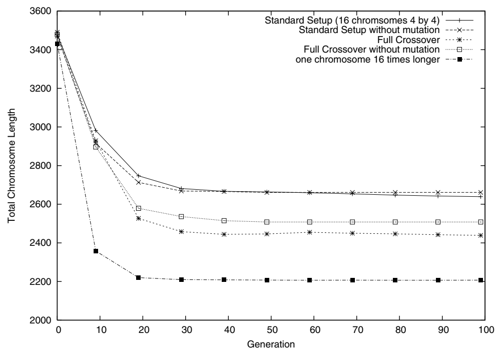

Figure 3: lengths of chromosomes throughout the run

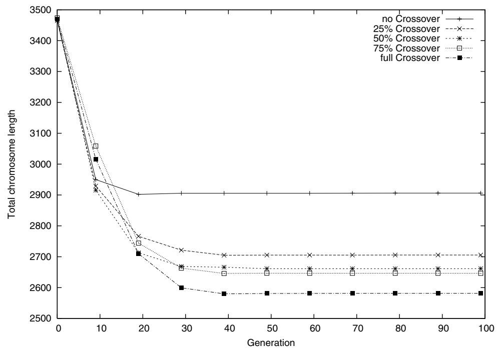

Figure 4: Increasing mutation rate speeds up the shortening of the length

NB: All results averaged over 20 runs

in an uncorrelated landscape.", but predicts that as average fitness increases this process will slow down drastically. This theory says nothing about why the overall trend of the *long jumps* is a decrease in length.

The shortening appears to be related to the amount of crossover which is allowed to occur within the system (see figure 4). The influence of crossover on length can be seen by taking a simple string one hundred bits long, with fifty 1's and fifty 0's. Two seemingly equivalent crossover operations occur to this string, one adds ten 1's the other takes away ten 0's. However, after the first operator, the fitness is $0.\overline{45}$, but in the second case the fitness is now $0.\dot{4}$. With tournament selection this means the second individual will have an advantage. This small difference in fitness would not last for long, since once the average number of 1's in the population grows then the chances of removing a block of 0's will drop. This fits with the observation that the length drops at the start of a run and then stabilises.

Interestingly, at the start, the numbers of ones and zeros in the chromosomes are roughly equal, at around 1750 of each. After twenty generations, the number of bits in the chromosomes has dropped to 2500 (see figure 3) and the percentage of zeros is now around 35%. This means that the best individuals in the population have lost approximately half the zeros they started off with, yet have only lost 7% of the ones with which they began.

Finally, shuffling must also play a role in the selection pressure, since when crossover is turned off the shrinkage still occurs. However, when shuffling is turned off by using only a single extra-long chromosome, the shrinkage occurs more, so shuffling cannot be a key source of selection pressure.

2.1 Conclusions

This paper looked at the dynamics involved in a simple variable length, multiple chromosome, genetic algorithm on the onemax problem and studied the effects of the operators within this system. It observed that the operators have varying effects at different stages of evolution. It has also looked at the evolution of length over the course of a run, finding that, in this case, there is an unexplained selection pressure at the start of a run to shorten the length.

Future work will include more studies into the observed selection pressure and looking at the dynamics on more complicated problems where gene linkage exists.

3. REFERENCES

[1] R. Cavill, S. Smith, and A. Tyrrell. Multi-chromosomal genetic programming. In *Proceedings of Gecco*, pages 1753–1761, 2005.

[2] R. Cavill, S. L. Smith, and A. M. Tyrrell. The performance of polyploid evolutionary algorithms is improved both by having many chromosomes and by having many copies of each chromosome on symbolic regression problems. In *Proceedings of CEC*, pages 935–941, 2005.

[3] E. D. de Jong and D. Thierens. Exploiting modularity, hierarchy, and repetition in variable-length problems. In *GECCO*, pages 1030–1041, 2004.

[4] I. Harvey. Species adaptation genetic algorithms: A basis for a continuing SAGA. In *Proc. of the First European Conference on Artificial Life*. MIT Press/Bradford Books, 1992.

[5] M. Nicolau and C. Ryan. Efficient crossover in the GAuGE system. In *EuroGP Proceedings*, volume 3003 of *LNCS*, pages 125–137, Coimbra, Portugal, 2004. Springer-Verlag.

[6] C. Ryan, J. J. Collins, and M. O Neill. Grammatical evolution: Evolving programs for an arbitrary language. In *EuroGP Proceedings*, LNCS volume 1391, pages 83–95, Paris, 14-15 1998. Springer-Verlag.

[7] C. Ryan, M. Nicolau, and M. O'Neill. Genetic algorithms using grammatical evolution. In *LNCS*, 2278:278–287, 2002.