Efficient Crossover in the GAuGE System

Miguel Nicolau and Conor Ryan

Biocomputing and Developmental Systems Group
Computer Science and Information Systems Department
University of Limerick, Ireland
{Miguel.Nicolau,Conor.Ryan}@ul.ie

Abstract. This paper presents a series of context-preserving crossover operators for the GAuGE system. These operators have been designed to respect the representation of genotype strings in GAuGE, thereby making sensible changes at the genotypic level. Results on a set of problems suggest that some of these operators can improve the maintenance and propagation of building blocks in GAuGE, as well as its scalability, and could be of use to other systems using structural evolving genomes.

1 Introduction

The GAuGE system (Genetic Algorithms using Grammatical Evolution) is a recently introduced position-independent genetic algorithm which, through a mapping process, maps a fixed length genotype string onto a fixed-length phenotype string, ensuring no under- or over-specification in the process. By encoding both the position and value of each phenotypic variable on genotype strings, GAuGE has the ability to structure these, to prioritise information, and to group together partial solutions, to minimise the chances of disrupting them.

Until now, a simple genetic algorithm has been used to generate binary strings that, through a mapping process, are interpreted as GAuGE strings, which in turn specify phenotype strings. A side effect of this is the ripple effect, by which a change of context for information exchanged between two individuals can lead to severe changes of its interpretation, at the phenotype level.

In this work, a set of crossover operators is presented, which are adapted to the GAuGE representation. These are context-preserving operators; by applying these, the context of information exchanged is the same, and therefore that information will keep its initial meaning. Experiments conducted suggest that, on the problems analysed, some of these operators scale better to problem difficulty, and are better able to discover, maintain and exchange partial solutions. These operators can therefore not only improve the performance of the GAuGE system, but also of other position-independent systems evolving the structure of genomes.

This paper is organised as follows. Section introduces the GAuGE system, and its relation to Grammatical Evolution, and includes a review of previous work and an example of the mapping process employed. Section introduces the newly designed crossover operators. Section describes the experiments conducted and the results obtained, while Section draws conclusions based on those results, and some lines of future research.

© Springer-Verlag Berlin Heidelberg 2004
2 GAuGE

The GAuGE system uses many of the biologically inspired features present in Grammatical Evolution (GE) [11], the main ones being a genotype to phenotype mapping (GPM), functional dependency between genes, and degenerate code.

In GE, a GPM process is used to map variable-length binary strings onto syntactically-correct programs, through the use of a grammar. Each binary string is mapped onto an integer string, through a transcription process, and then, through a translation process, those integers are used to choose productions from the grammar, to create a program (phenotype). This process, based on the analogous process in molecular biology, provides a division between the search space (binary strings) and the solution space (evolved programs) [2].

In GAuGE, a similar process is employed. A population of (fixed length) binary strings is created in the genotypic space, and each of these is also transcribed onto an integer string. This integer string is then interpreted as a sequence of (position, value) specifications that, through a mapping process, generate a fixed-length phenotype string, which is neither under nor over-specified.

In GE, the function of a gene can affect the function of the genes that follow it. Indeed, the rule from which a gene is used to choose a production depends on the mapping process up to that point; this means that if the functionality of a gene changes, the functionality of subsequent genes is likely to change as well.

This feature, called functional dependency, is present in GAuGE as well. Each position specification across a genotype string is dependent on previous specifications, in order to create a fully specified phenotype string.

Finally, the use of degenerate code plays an important role in GE: by using the mod operator to map an integer to a choice of productions from a grammar rule, neutral mutations can take place [7], creating a many-to-one mapping between the search and solution spaces, and introducing variety at the genotypic level.

In GAuGE, this feature is also present, as a direct result of the mapping process employed. It has also been shown that the explicit introduction of degeneracy can reduce structural bias at the genotypic level [9].

2.1 Previous Work

Previous work has used similar techniques as the ones employed in GAuGE. Some of Bagley’s [1] simulations used an extended representation to encode both the position and the value of each allele. Some reordering operators were also designed [10], which combine inversion and crossover operators on extended representations. Later on, the so-called messy genetic algorithms [5] applied the principle of separating the gene and locus specifications with considerable success, and have since been followed by many competent GAs.

The Random Keys Genetic Algorithm [3] introduced a mapping process that ensures an error-free and fully specified sequence of ordinal numbers. More recently, Harik [6] applied the principles of functional dependency in the Linkage Learning Genetic Algorithm, in which the functionality of a gene is dependent on a chosen interpretation point, and the genes between that point and itself.