Evaluating GP Schema in Context

Hammad Majeed, Conor Ryan and R. Muhammad Atif Azad
Biocomputing and Developmental Systems Group
Computer Science and Information Systems Department
University of Limerick, Ireland.
{hammad.majeed, conor.ryan, atif.azad}@ul.ie

ABSTRACT

We propose a new methodology to look at the fitness contributions (semantics) of different schemata in Genetic Programming (GP). We hypothesize that the *significance* of a schema can be evaluated by calculating its *fitness contribution* to the total fitness of the trees that contain it, and use our methodology to test this hypothesis.

It is shown that this method can also be used to identify schemata that are important in terms of both individual runs and individual problems (that is, schema that will be important across many runs on a particular problem).

The usefulness of this study to existing schema theories and its *effective* use in the detection of *introns*, in the identification of potentially useful modular functions are also discussed in this paper.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Genetic Programming*

General Terms

Algorithms, Theory

Keywords

Tree Semantics, Module Acquisition, Schema Theory

1. INTRODUCTION

The main difficulty with analysing GP systems is associated with their use of variable sized tree structures. Unlike a GA's simple linear string chromosome, a GP tree is a complex structure with many inherent complexities. Two prominent ones are *sub-tree context* and *tree fragment evaluation*. That is, each sub-tree within a parent tree has some context and it is not possible to evaluate the sub-tree independent of the parent tree. This makes it impossible to look at the fitness of a *fragment* of schema involving an incomplete tree.

GECCO'05, June 25–29, 2005, Washington, DC, USA.
ACM 1-59593-010-8/05/0006.

This paper presents a new approach to *explicitly* look at the fitness contributions of the schemata within a tree. Currently, it is a two step offline process. In the first step, a schema is selected from the final generation and compared with the entire populations of all the preceding generations. Every time an individual matches the schema, its fitness is noted. Then, the subtree instantiating this schema is replaced by an *intron* and the individual is re-evaluated. The difference between the two fitness values gives an idea of the schema's contribution in the original individual.

2. EXAMINING SCHEMA IN CONTEXT

We are concerned with investigating the significance of a schema with respect to its position (depth) and size (nodes) within a tree, referred to below as the *container-tree*. It is widely accepted by GP community that a node (or sub-tree) closer to the root node is more significant than the nodes found below it. This is because it dictates the meaning to the lower nodes. It can also be argued that the size of the schema may have effect on its fitness contributions, as in general the bigger trees contribute more towards the overall fitness of the container-tree.

For this study, a schema is a subtree selected from the final generation and has to fulfill the following criteria:

(i) it should be present in at least half of the population.

(ii) its own depth should at least be equal to a minimum provided depth. As this was an exhaustive search, all legal depths for each tree were examined.

Candidate schemata were searched at all depths within the population, starting from 1 to the maximum allowed depth within the container-tree. This is an exhaustive search and ensures selection of all possible schemata. This, however, can be varied, to search only for schema above or below a particular depth.

2.1 Generalization of Schema

After the selection of a schema, its nodes are replaced probabilistically with # (don't care) nodes with bias towards the lower nodes. To replace a node, a roulette wheel was constructed. It helps to mark a depth and then a node for replacement. In the generalization step all the constants in the selected schema are replaced with #. This avoids match failures due to different constants (which is a common observation). Note, replacement of the constants in the selected schema does not stop its # node from matching a

constant value. This step makes this study more comprehensive by exploring larger space of instances for the selected schema.

2.2 Contribution of Schema

The contextual analysis in this paper is made possible by calculating the fitness contribution of any sub-tree within a tree. This is a three step process. In the first step, an individual containing the sub-tree is evaluated, before the sub-tree is replaced with an *identity* node. The identity node acts as an intron in the container-tree and cancels the effect of the sub-tree. After replacement, the individual is then re-evaluated. The difference of the two fitnesses is the contribution of the sub-tree in the container-tree

The working principle of an identity node is similar to the identity function in set theory. In our definition, an identity node always replaces some node or sub-tree. This replacement cancels out the effect of the replaced node or sub-tree.

3. EXPERIMENTS

To test the ability of our method to identify useful schemata, we conducted experiments using Koza's quartic polynomial symbolic regression problem, and a population of size 500 was allowed to evolve for 50 generations. Sub-tree crossover with probability 0.9 and reproduction with probability 0.1 were used. The initial generation was generated using the *ramped half and half* method. The initial tree depth varied from 2-6 while the maximum depth of the trees was set to 17, and 100 independent runs were conducted to note the trend of selected schemata across different runs. The same function set as that employed by Koza was used.

We examined three different measurements for this study. The first was to note the **fitness contribution** of the selected schema in a run. This was calculated for each generation by averaging the contribution of all instances of a schema in that particular generation. Similarly, the **size contribution** of each of the identified schema was noted, and averaged in the same way, while the third measurement, **schema depth**, tracked the average depth of each schema.

3.1 Schema contribution to fitness and size

The first set of experiments look at the correlation between size and fitness of different schemata. The experimental results show that there are schemata with fitness contributions directly propotional to their size contributions towards the container-trees. We mark them as *significant* for the run. Over the course of evolution their instances grow in size and fitness. On the other hand, there are other schemata which are contributing 90% towards the size of container-trees but still their fitness contributions are insignificant. This behavior can be due to the *incorrect* contexts of the instances of the schema. Another possible explanation is that they are acting as *introns* or are *part* of an intron sub-tree(see section 4.2 for details). This shows that the fitness contribution of a schema is *not just a function of its size contributions* but also the *correct context of the sub-trees.*

3.2 Comparing schemata within runs

Next we looked at comparing the different schemata found in a run. Most of the schemata found in the runs show a similar behavior, i.e. their contributions drop with time. The longer schemata fail to have any instances in the initial generations but, once found, their fitness contributions either remain constant or increase with time. Note the fitness contributions of schemata can be *negative*. A schema has a negative fitness contribution when its removal results in an *increase* of fitness of the container-tree.

4. BENEFIT OF THIS STUDY

This section discusses the possible uses and benefits of the proposed approach along with it's effect on the existing schema theories.

4.1 Modules discovery and Use

This method can be applied online to identify semantically significant schemata from a population, *given their current context*. This information can then be used to generate better individuals for subsequent generations. It can be done either by only allowing the individuals containing the significant schemata to propagate to the next generation or by generating new instances of a useful schema and introducing them to the subsequent generation(s). This could also be of enormous benefit to module acquisition strategies such as ARL [3], Iba's [2] guided recombination and Angeline's MA [1], the performance of all of which depend to a large extent on the successful identification of useful subtrees. Using our technique, a table of *good* schemata could be constructed for each generation, which can then be consulted when selecting a sub-tree for encapsulation. Sub-trees that are instances of schemata that appear in this table could then be made more likely to be selected.

4.2 Identification of dead code

The method described in this paper has been shown to identify the fitness contribution of a block towards an individual. It estimates the value of a subtree *in context* of a particular tree. Iba and de Garis [2] also calculate the worth of the constituent subtrees of an individual by treating each subtree as an independent program. This evaluation does not inform us about the worth of a subtree towards the main tree.

More importantly, code that is neutral towards the fitness of an individual can be identified with our technique. For example consider a schema (+ # (* (- X X) #)). The subtree (* (- X X) #) acts as an intron by virtue of multiplication of (- X X) with any expression #. The fitness contribution of the entire right hand side is exactly 0.0.

5. REFERENCES

[1] Peter John Angeline. Genetic programming and emergent intelligence. In Kenneth E. Kinnear, Jr., editor, *Advances in Genetic Programming*, chapter 4, pages 75–98. MIT Press, 1994.

[2] Hitoshi Iba and Hugo de Garis. Extending genetic programming with recombinative guidance. In Peter J. Angeline and K. E. Kinnear, Jr., editors, *Advances in Genetic Programming 2*, chapter 4, pages 69–88. MIT Press, Cambridge, MA, USA, 1996.

[3] Justinian P. Rosca and Dana H. Ballard. Discovery of subroutines in genetic programming. In Peter J. Angeline and K. E. Kinnear, Jr., editors, *Advances in Genetic Programming 2*, chapter 9, pages 177–202. MIT Press, Cambridge, MA, USA, 1996.