Examining Mutation Landscapes In Grammar Based Genetic Programming

Eoin Murphy Michael O'Neill
Anthony Brabazon

Natural Computing Research and Applications Group,
Univeristy College Dublin, Ireland.
{eoin.murphy,m.oneill,anthony.brabazon}@ucd.ie

Abstract

Representation is a very important component of any evolutionary algorithm. Changing the representation can cause an algorithm to perform very differently. Such a change can have an effect that is difficult to understand. This paper examines what happens to the grammatical evolution algorithm when replacing the commonly used context-free grammar representation with a tree-adjunct grammar representation. We model the landscapes produced when using integer flip mutation with both representations and compare these landscapes using visualisation methods little used in the field of genetic programming.

1 Introduction

Three very important components of any evolutionary algorithm are the representation, the variation operations and the fitness function. The interaction of these components within the algorithm forms a complex process and the modification of any one of them can have a major effect on how the algorithm performs. Such an effect may not be immediately obvious and is difficult to understand. Koza and Poli [12] said that visualising the program search space would be useful and help us understand how the algorithm operates.

Grammatical Evolution (GE) [8, 21] has recently been extended to make use of tree-adjunct grammars (TAG) [10, 11] in place of the usual grammar type, context-free grammars (CFG) [16]. TAGs have shown promise in the field of Genetic Programming (GP) [5, 6, 7, 17] as well as other fields in natural computing [1]. This promise carried over when TAGs were incorporated into GE, i.e., Tree-Adjunct Grammatical Evolution (TAGE), in the form of an increased ability to find fitter solutions in fewer generations and an increased success rate [16]. Previous work has examined how the TAG representation overcomes some

of the structural difficulties present in GP [8], but the full extent of how TAGs affect GE is unclear.

Landscapes are a tool to help understand complex processes [9]. They have been employed here in an attempt to further understand how the use of TAGs in GE affects performance. Using a single variation operation, Integer Flip Mutation (IFM), the landscapes of a number of different problems are examined for both TAGE and GE. The IFM operation is where the integer value of a codon is replaced with a new random integer value. Viewing the entire search space/landscape is difficult due to its large size and high complexity. To alleviate this problem, this study employs a method of visualisation little used in the field of GP, heat maps.

This paper compares the single IFM landscapes of GE and TAGE for a series of problems in an attempt to further understand how the change in representation affects each algorithm's ability to search.

This section is followed by a brief introduction to the landscape model used in this study in section 2 along with a description of GE and TAGE in sections 3 and 4; The experiments performed are outlined in section 5 along with the results; These are followed by a discussion in section 6 and some conclusions and future work in the final section.

2 Landscapes

The landscape model used in this paper is as defined by Jones [9], where he quotes Nilsson [18], *"In its broadest sense, problem solving encompasses all of computer science because any computational task can be regarded as a problem to be solved."*, Pearl [22], *"Every problem-solving activity can be regarded as the task of finding or constructing an object with given characteristics"*, and Rich [23], *"Every search process can be viewed as a traversal of a directed graph in which each node represents a problem state and each arc represents a relationship between the states represented by the nodes it connects"*, stating that from the above statements one can conclude that search is ubiquitous and that it can be described as a process on a graph structure [9]. It is for this reason that he adopts a graph as a view of his landscape model.

The full description of the landscape model is outlined at length in [9]. It is sufficient to say for this study that the landscape model can be written as

$$L = (R, \phi, f, F, >_F) \tag{1}$$

where R is the representation space, ϕ is the operator (in this case a genetic operator), the function f which maps a multi-set of R, $M(R){:}\mapsto F$ for some set F, the fitness space, and a partial ordering $>_F$ over F. The landscape, L can be viewed as a directed labeled graph where the set of vertices, V, is a subset of $M(R)$ and an edge exists between the vertex v and the vertex w if $p(\phi(v, w)) > 0$.

In this study the landscapes are defined using the space of chromosomes paired with either a CFG or TAG as R. The object space, O, is the solu-

tion/phenotypic space. ϕ is the IFM operator, and f is the fitness function. The landscapes can be viewed as graph structures both where $V \subseteq M(R)$, and where $V \subseteq O$ (each vertex is a phenotype, but edges are dependent on ϕ and R).

3 Grammatical Evolution

GE is a grammar-based approach to GP, combining aspects of Darwinian natural selection, genetics and molecular biology with the representational power of grammar formalisms [2, 15, 21]. The grammar, written in Backus-Naur form, enables GE to define and modify the legal expressions of an arbitrary computer language. Moreover, the grammar also enables GE to modify the structure of the expressions generated, something that is not trivial for other forms of GP. In addition, the separation of the genotype from the phenotype in GE allows genetic operations to be applied not only to the phenotype, as in GP, but also to the genotype, extending the search capabilities of GP. GE is considered to be one of the most widely applied GP methods today [15].

3.1 Grammatical Evolution by Example

Representation in GE consists of a grammar and a chromosome, see Fig. 1. A genotype-phenotype mapping uses values from the chromosome to select production rules from the grammar, building up a derivation tree. The phenotype can be extracted from this tree's frontier.

The mapping begins with the start symbol, `<e>`. The value of the first codon, `12`, is read from the chromosome. The number of production rules for the start symbol are counted, `2`, `<e><o><e>` and `<v>`. The rule to be chosen is decided according to the mapping function `i mod c`, where `i` is the current codon value and `c` is the number of choices available, e.g, `12 mod 2 = 0`, therefore the zero-th rule is chosen. `<e>` is expanded to `<e><o><e>`. This expansion forms a partial derivation tree with the start symbol as the root, attaching each of the new symbols as children. The next symbol to expand is the first non-terminal leaf node discovered while traversing the tree in a depth first manner. However, it should be noted that there is on-going study into variations on the method used to choose which node to expand next [19, 20]. In this case the left-most `<e>` is chosen. The next codon, `3`, is read, expanding this `<e>` to `<v>` and growing the tree further. The next symbol, `<v>` is expanded using the next codon, `7`. `7 mod 2 = 1`, so the rule at index 1, `Y`, is chosen.

Derivation continues until there are no more non-terminal leaf nodes to expand, or until the end of the chromosome has been reached. If there are non-terminal leaf nodes left when the end of the chromosome has been reached, derivation can proceed in one of a few different manners. For example, a bad fitness can be assigned to the individual, so it is highly unlikely that this individual will survive the selection process. Alternatively the chromosome can be wrapped, reusing it a predefined number of times. If after the wrapping limit

```
Grammar:
<e>:= <e><o><e> | <v>
<o>:= + | -
<v>:= x | y

Chromosome:
12, 3, 7, 15, 9, 10, 14
```

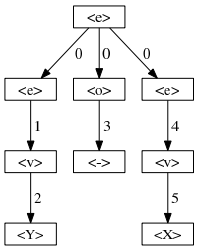


Figure 1: Example GE grammar, chromosome and resulting derivation tree.

has been reached and the individual is still invalid, it could then be assigned a bad fitness. The complete derivation tree for this example is shown in Fig. 1.

4 Tree-Adjunct Grammatical Evolution

TAGE, like GE, uses a representation consisting of a grammar and a chromosome. The type of grammar used in this case is a TAG rather than a CFG. A TAG is defined by a quintuple (T, N, S, I, A) where:

1. T is a finite set of terminal symbols;
2. N is a finite set of non-terminal symbols: $T \cap N = \emptyset$;
3. S is the start symbol: $S \in N$;
4. I is a finite set of finite trees. The trees in I are called *initial trees* (or α trees). An initial tree has the following properties:
   - the root node of the tree is labeled with S;
   - the interior nodes are labeled with non-terminal symbols;
   - the leaf nodes are labeled with terminal symbols;
5. A is a finite set of finite trees. The trees in A are called *auxiliary trees* (or β trees). An auxiliary tree has the following properties:
   - the interior nodes are labeled with non-terminal symbols;
   - the leaf nodes are labeled with terminal symbols apart from the foot node which is labeled with the same non-terminal symbol as the root; the convention in [10] is followed and foot nodes are marked with *.

An initial tree represents a minimal non-recursive structure produced by the grammar, i.e., it contains no recursive non-terminal symbols. Inversely, an auxiliary tree of type X represents a minimal recursive structure, which allows recursion upon the non-terminal X [14]. The union of initial trees and auxiliary trees forms the set of *elementary trees*, E; where $I \cap A = \emptyset$ and $I \cup A = E$.

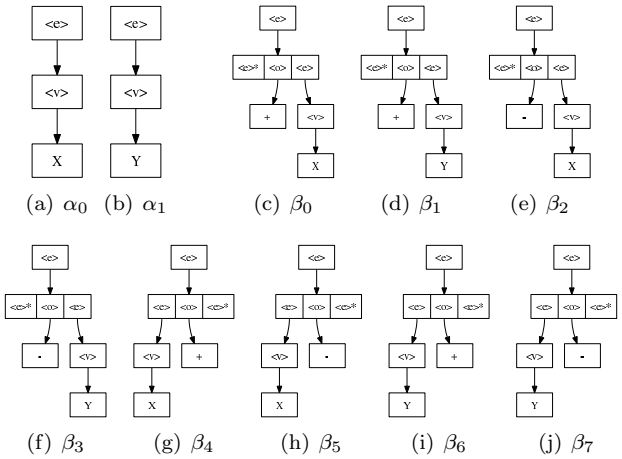


Figure 2: Initial and auxiliary tree sets of the TAG produced from the CFG in Fig. 1.

During derivation, composition operations join elementary trees together. The adjunction operation takes an initial or derived tree a, creating a new derived tree d, by combining a with an auxiliary tree, b. A sub-tree, c is selected from a. The type of the sub-tree (the symbol at its root) is used to select an auxiliary tree, b, of the same type. c is removed temporarily from a. b is then attached to a as a sub-tree in place of c and c is attached to b by replacing c's root node with b's foot node. An example of TAG derivation is provided in Section 4.1.

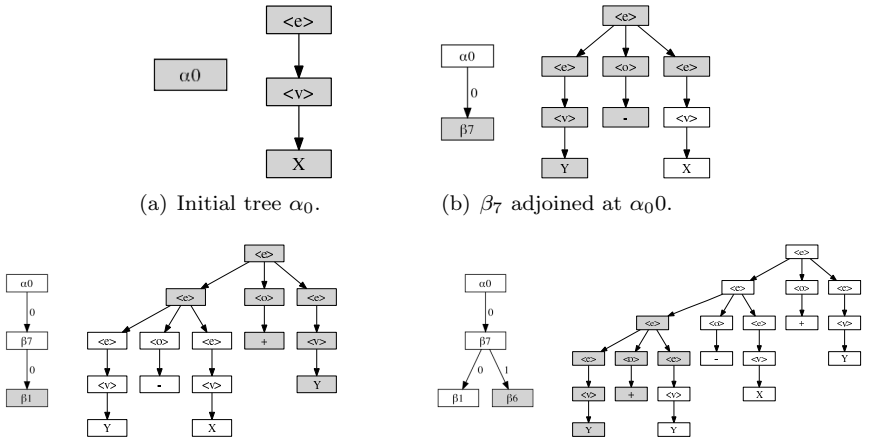
4.1 Tree-Adjunct Grammatical Evolution by Example

TAGE generates TAGs from the CFGs used by GE. Joshi and Schabes [10] state that for a *"finitely ambiguous CFG*[1] *which does not generate the empty string, there is a lexicalised tree-adjunct grammar generating the same language and tree set as that CFG"*. An algorithm was provided by Joshi and Schabes [10] for generating such a TAG. The TAG produced from Fig. 1 is shown in Fig. 2.

Derivation in TAGE is different to GE. A TAGE derivation tree is a tree of trees. That is to say a node in a TAGE derivation tree contains an elementary tree. The edges between those nodes are labeled with a node address of the tree in the parent derivation node. It is at this address that the beta tree in the child node is to be adjuncted. The derived tree in TAGE is a tree of symbols, similar to GE's derivation tree, resulting from the application of the adjunction operations defined in the TAGE derivation tree.

Given the TAG G, where $T = \{x, y, +, -\}$, $N = \{< e >, < o >, < v >\}$, $S =< e >$ and I and A are shown Fig. 2, derivation, using the chromosome

[1] A grammar is said to be *finitely ambiguous* if all finite length sentences produced by that grammar cannot be analysed in an infinite number of ways.



(a) Initial tree α_0.

(b) β_7 adjoined at $\alpha_0 0$.

(c) β_1 adjoined at $\beta_7 0$.

(d) β_6 adjoined at $\beta_7 1$.

Figure 3: The derivation tree and corresponding derived tree at each stage of derivation in TAGE. The shaded areas indicate the new content added to the tree at each step.

from Fig. 1, operates as follows. An initial tree is chosen to start derivation. The first codon value, `12`, is read and is used to choose an initial tree based on the number of trees in I. Using the same mapping function as GE, `12 mod 2 = 0`, the zero-th tree, α_0, is chosen from I. This tree is set as the root node of, `t`, the derivation tree, see Fig. 3(a).

Next we enter the main stage of the algorithm. A location to perform adjunction must be chosen. The set `N` is created of the adjunct-able addresses available within all nodes(trees) contained within `t`. An adjunct-able address in a tree is the breadth first traversal index of a node labeled with a non-terminal symbol of which there is an auxiliary tree of that type, and there is currently no auxiliary tree already adjoined to the tree at that index. In this case `N =` $\{\alpha_0 0\}$, so a codon is read and an address is selected from `N`, `3 mod 1 = 0` indicating which address to choose, `N[0]`. Adjunction will be performed at $\alpha_0 0$, or index `0` of tree α_0, `<e>`. An auxiliary tree is now chosen from A that is of the type `l`, i.e., the label of it's root node is `l`, where `l` is the label of the node adjunction is being performed at. In this case `l = <e>`. Since there are 8 such trees in A, `7 mod 8 = 7`, β_7 is chosen. This is added to `t` as a child of the tree being adjoining to, labeling the edge with the address `0`, see Fig. 3(b). The adjunct-able addresses in β_7 will be added to `N` on the next pass of the algorithm. This process is repeated until all remaining codons have been read. The resulting derivation and derived trees at each stage of this process can be seen in Fig. 3.

5 Experiments and Results

The aim of this study is to compare GE and TAGE IFM landscapes in order to ascertain some insight into how TAGE improves the algorithm's performance. In order to compare landscapes bounds must be set on the size of the landscapes. Since the size and form of solutions are rarely known a priori, the grammars used in GE tend to be recursive. As a result the structural space of possible solutions is infinite, and hence the landscape is infinite, restricted only by the number of codons available to the mapping procedure. This applies to both TAGE and CFG since they generate the same language.

5.1 Experimental Setup

In order to restrict the landscapes a specific number of codons, N, is selected as the maximum length of a TAGE chromosome. A value for N is chosen for each problem examined in order to sufficiently restrict the size of the landscapes. At each chromosome length, from one to N, an enumeration of all possible chromosomes is performed, building up the representation space, R. It is required for TAGE to process each increasing length of chromosome since with mutation alone, the number of codons used when mapping cannot change and hence TAGE would not be able to represent the same set of phenotypes as GE.

The enumeration is performed by initially selecting a chromosome of all zeros. At each position along the chromosome, every possible IFM is independently performed. That is to say, the mapping procedure is stopped at the each codon and the total number of possible choices at that codon is counted. This indicates how many different IFMs can be applied at each codon, creating the set of all chromosomes one IFM away from the original. Each of these neighbouring chromosomes are mapped, if both the original and the neighbour is valid, i.e., if the chromosome maps to an executable solution (for TAGE this is not an issue, since all chromosomes are valid), an edge/connection is recorded between them. If the neighbour has never been observed, it is added to a set of chromosomes from which new initial chromosomes are drawn to repeat this process.

Once this set of chromosomes is depleted, the chromosome length is incremented and the process repeated with a new initial chromosome. The process halts when the all chromosomes of length N have been processed.

The resulting phenotypes are used to repeat the above process for GE. Rather than setting a chromosome length limit, the length is incremented until the set of phenotypes generated contains the set of phenotypes generated by TAGE.

5.2 Problems

Standard GE was compared to TAGE using four classic benchmark problems taken from the GP literature. The CFGs used by GE and to generate TAGs for each problem are shown in Fig. 4.

```
Even-5 parity grammar:
<prog>   ::= <expr>
<expr>   ::= <expr> <op> <expr>
         | ( <expr> <op> <expr> )
         | <var>
         | <pre-op> ( <var> )
<pre-op> ::= not
<op>     ::= "|" | & | ^
<var>    ::= d0 | d1 | d2 | d3 | d4
```

```
Max grammar:
<prog> ::= <expr>
<expr> ::= <op><expr><expr>
         | <var>
<op>   ::= + | *
<var>  ::= 0.5
```

```
Symbolic Regression grammar:
<expr> ::= (<op><expr><expr>)
         | <var>
<op>   ::= + | - | *
<var>  ::= x0 | 1.0
```

```
Six Multiplexer grammar:
<B> ::= (<B>)&&(<B>)
      | (<B>)"||"(<B>)
      | !(<B>)
      | (<B>) ? (<B>) : (<B>)
      | a0 | a1 | d0| d1 | d2 | d3
```

Figure 4: CFG grammars in Backus-Naur form used for all the benchmark problems.

Even-5-parity: The five input even-parity boolean function, in which the best fitness is obtained when the correct output is returned for each of the 2^5 test cases. A value of 3 was used for N.

Symbolic Regression: The classic quartic function, $x + x^2 + x^3 + x^4$. Fitness is the sum of the error across 20 test cases drawn from the range $[-1, 1]$. Successful solutions have an error less than 0.01, as described in [13]. A value of 5 was used for N.

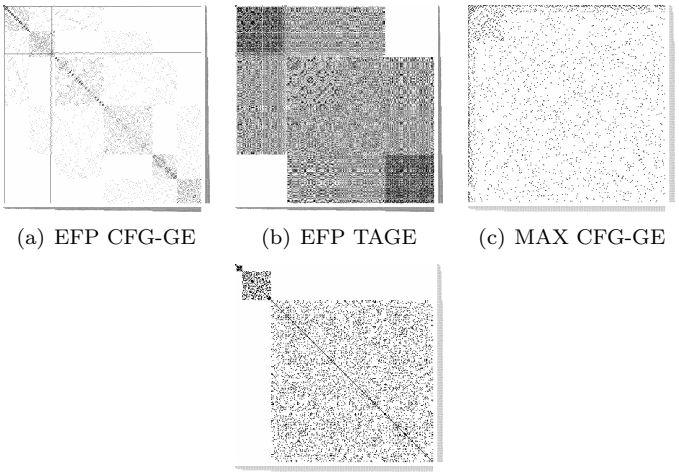
Six Multiplexer: The classic GP two input and four output line boolean function. Fitness is measured by how many of the 64 test cases generate correct outputs. A value of 3 was used for N.

Max: This problem, as described in [3], aims to evolve a tree whose growth is constrained by a depth limit, that when the tree's phenotype is executed, returns the largest value possible. A function set of $\{+, *\}$ and a terminal set of $\{0.5\}$ are used. A max tree depth of 8 was used for the purposes of these experiments. A value of 9 was used for N.

5.3 Visualisations

Viewing the landscapes as 2D graphs is not feasible due to their large size and high complexity. 2D heat maps are used instead to map the connections in the landscape. Heat maps are little used in GP literature and are an effective way of graphically representing data in a 2 dimensional map, where the values of the variable being visualised are represented as colours.

Rather than using the genotypic landscape, i.e., where each vertex represents a single genotype from the representation, the phenotypic landscape is used, since comparing the genotypes of two different representations may not be useful and both representations in question generate the same phenotypic space.



(a) EFP CFG-GE (b) EFP TAGE (c) MAX CFG-GE

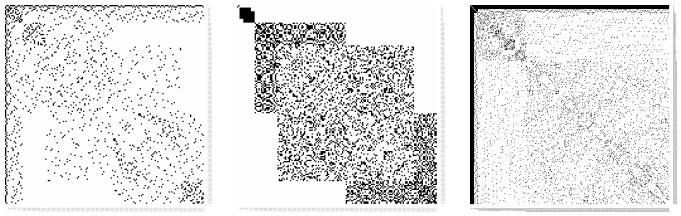
(d) MAX TAGE

Figure 5: Connection Maps: Even Five Parity (a) (b) for a max TAGE chromosome length of 3; Max (c) (d) for a max TAGE chromosome length of 9.

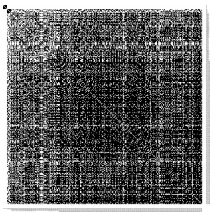
Connection Maps (CM) are heat maps where the set of commonly generated phenotypes label each axis. If the genotypes of two phenotypes are one IFM away, the shared cell is marked. CMs give insight into how well connected each phenotype is within the landscape. The denser the CM, the greater the representation's ability to move from one phenotype to another.

The CMs for both setups for each of the problems can be seen in Figs. 5 and 6. The axes of these figures are labeled with the phenotypes in ascending order of length, from the top left to the bottom right.

Frequency Maps aim to address one of problems with the CMs described above. CMs do not take into account that there may be more than one connection between two phenotypes. This can occur due to GE and TAGE having redundant mappings. However, neutral mutation was not allowed in this study. The frequency of connections between phenotypes is important since if one connection from a phenotype has a high frequency and all of the other connections from that phenotype have a relatively low frequency of connections then there is a much higher probability that a mutation will follow the connections of high frequency. Frequency maps colour each cell from 25% grey (0) to red (200+) depending on the cell's degree of connectivity. The upper bound of 200 connections was to ensure a feasible colour delta when colour coding the maps due to the large number of relatively low frequency cells and a small number of much higher frequency cells. Frequency maps for only the symbolic regression and six multiplexer problems can be seen in Fig. 7 due to space restrictions.

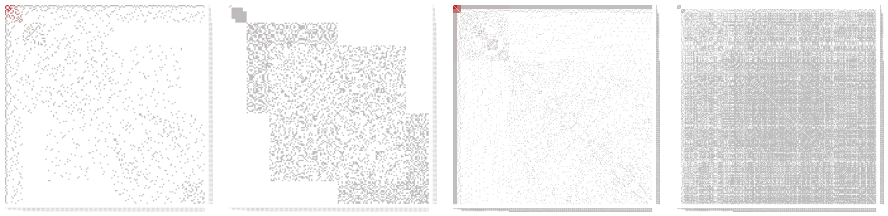


(a) SR CFG-GE (b) SR TAGE (c) SM CFG-GE



(d) SM TAGE

Figure 6: Connection Maps: Symbolic Regression (a) (b) for a max TAGE chromosome length of 5; Six Multiplexer (c) (d) for a max TAGE chromosome length of 3.



(a) SR CFG-GE (b) SR TAGE (c) SM CFG-GE (d) SM TAGE

Figure 7: Frequency Maps: Symbolic Regression (a) (b); Six Multiplexer (c) (d).

6 Discussion

The CMs in figures 5 and 6 show that phenotypes in the TAGE landscapes, across the problems/grammars examined, are much more connected than the same phenotypes in the GE landscapes. This might not necessarily improve TAGE's ability to move from one phenotype to another of better fitness since the concept of fitness is not present in the CM plots. It was however noted by Murphy et al. [16] that TAGE maintains a much larger fitness variance within the population than GE. It was suggested that this variance, as a result of a more diverse population, could help TAGE avoid getting stuck at local optima [16]. The high degree of connectivity visible here could be attributed with helping to increase diversity within the population.

Interestingly, it can also be seen that mutation alone is not sufficient for TAGE to explore the entire search space. Unlike GE where an IFM can reduce the effective size of the chromosome, TAGE makes use of the entire chromosome and as a result IFM cannot change the size of a TAGE derivation tree. In order to enable a full exploration of the search space additional operators capable of changing the length of the chromosome would be needed. This would be as simple as a codon insertion/deletion operator or more complex such as one point crossover [16]. The clusters of connections in the top left corner of each of the TAGE sub-figures are the connections between the shorter chromosomes generated during setup, the remainder of the cells are white due to the lack of connections with the phenotypes of the larger chromosome.

Furthermore, the frequency maps in Fig. 7 show that in GE, the phenotypes produced from a smaller amount of the chromosome have a disproportionately high frequency of connections amongst themselves (see the red cells in the top left corner of (a) and (c)), and to a lesser extent with the rest of the phenotypes (left and top borders of (a) and (c)). In some cases the frequency of connections of these cells are orders of magnitude greater than the frequency of connections of the larger phenotypes. This indicates that the CFGs used in this study have a bias towards shorter phenotypes. A bias that doesn't appear in the frequency maps of TAGE's landscapes. This feature of TAGE may help avoid some of the initialisation problems experienced by GE outlined by Harper [4]. For example, when the grammar is dermined to be *explosive* randomised individuals tend to be short having a lasting effect on the algorithms performance.

7 Conclusions

IFM landscapes were generated for a number of problems using both CFG-based GE and TAGE. Viewing an entire landscape directly is very difficult [12]. As such, the landscapes were restricted in size, and a number of different plots were employed to enable indirect analysis and comparison of the landscapes.

For the problems and grammars used in this study, it was found that phenotypes in the TAGE landscapes have a much higher degree of connectivity to the rest of the phenotypes than their counterparts in the GE landscapes. This may

help explain the increased diversity within TAGE populations observed previously. Moreover, it was discovered that the connectivity in the TAGE landscapes is much more evenly distributed between the other phenotypes in the landscape. Whereas in the GE landscape, shorter phenotypes are much more densely connected not only between themselves, but also, to a lesser extent, to the rest of the landscape.

This study presented a method for comparing large and highly complex landscapes using specific visualisation methods. This method of comparison can not only be further applied to the field of GE, but also to broader fields such as GP and genetic algorithms. Such an extension might enable better comparisons of each of the fields for a given problem, e.g., GP versus GE.

Future work arising from this study includes extending the method to other operators, allowing a better comparison of GE and TAGE; incorporating fitness into the CM method; and as mentioned above, comparing other representations with both GE and TAGE.

Acknowledgments

This research is based upon works supported by the Science Foundation Ireland under Grant No. 08/IN.1/I1868.

References

[1] H. Abbass, N. X. Hoai, and R. I. (Bob) McKay. AntTAG: A new method to compose computer programs using colonies of ants. In *Proceedings, 2002 World Congress on Computational Intelligence*, volume 2, pages 1654–1666. IEEE Press, 2002.

[2] Ian Dempsey, Michael O'Neill, and Anthony Brabazon. *Foundations in Grammatical Evolution for Dynamic Environments*. Studies in Computational Intelligence. Springer, 2009.

[3] C. Gathercole and P. Ross. An adverse interaction between crossover and restricted tree depth in genetic programming. In *GECCO '96: Proceedings of the First Annual Conference on Genetic Programming*, pages 291–296, Cambridge, MA, USA, 1996. MIT Press.

[4] Robin Harper. GE, explosive grammars and the lasting legacy of bad initialisation. In *IEEE Congress on Evolutionary Computation (CEC 2010)*, Barcelona, Spain, 18-23 July 2010. IEEE Press.

[5] N. X. Hoai. Solving the symbolic regression with tree-adjunct grammar guided genetic programming: The preliminary results. In *Australasia-Japan Workshop on Intelligent and Evolutionary Systems*, University of Otago, Dunedin, New Zealand, 19-21st November 2001.

[6] N. X. Hoai, R. I. McKay, D. Essam, and R. Chau. Solving the symbolic regression problem with tree-adjunct grammar guided genetic programming: The comparative results. In *Proceedings of the 2002 Congress on Evolutionary Computation CEC2002*, pages 1326–1331. IEEE Press, 12-17 May 2002.

[7] Nguyen Xuan Hoai, R. I. McKay, and H. A. Abbass. Tree adjoining grammars, language bias, and genetic programming. In *Genetic Programming, Proceedings of EuroGP'2003*, volume 2610 of *LNCS*, pages 335–344, Essex, 14-16 April 2003. Springer-Verlag.

[8] Nguyen Xuan Hoai, R. I. (Bob) McKay, and Daryl Essam. Representation and structural difficulty in genetic programming. *IEEE Transactions on Evolutionary Computation*, 10(2):157–166, April 2006.

[9] Terry Jones. *Evolutionary Algorithms, Fitness Landscapes, and Search*. PhD thesis, University of New Mexico, 1995.

[10] A.K. Joshi and Y. Schabes. Tree-Adjoining Grammars. *Handbook of Formal Languages, Beyond Words*, 3:69–123, 1997.

[11] A.K. Joshi, L.S. Levy, and M. Takahashi. Tree adjunct grammars. *Journal of Computer and System Sciences*, 10(1):136–163, 1975.

[12] John R. Koza and Riccardo Poli. Genetic programming. In *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, chapter 5, pages 127–164. Springer, 2005.

[13] J.R. Koza. *Genetic Programming*. MIT Press, 1992.

[14] A. Kroch and A.K. Joshi. The Linguistic Relevance of Tree Adjoining Grammar, Technical Report, University Of Pennsylvania, 1985.

[15] Robert McKay, Nguyen Hoai, Peter Whigham, Yin Shan, and Michael ONeill. Grammar-based genetic programming: a survey. *Genetic Programming and Evolvable Machines*, 11:365–396, 2010.

[16] Eoin Murphy, Michael O'Neill, Edgar Galvan-Lopez, and Anthony Brabazon. Tree-adjunct grammatical evolution. In *2010 IEEE World Congress on Computational Intelligence*, pages 4449–4456, Barcelona, Spain, 18-23 July 2010. IEEE Computational Intelligence Society, IEEE Press.

[17] X. H. Nguyen and R. I. (Bob) McKay. A framework for tree-adjunct grammar guided genetic programming. In *Post-graduate ADFA Conference on Computer Science*, pages 93–100, Canberra, Australia, 2001.

[18] Nils J. Nilsson. *Problem-Solving Methods in Artificial Intelligence*. McGraw-Hill Pub. Co., 1971.

[19] M. O'Neill, A. Brabazon, M. Nicolau, S.M. Garraghy, and P. Keenan. πGrammatical Evolution. In *Genetic and Evolutionary Computation GECCO 2004*, pages 617–629. Springer Berlin / Heidelberg, 2004.

[20] M. O'Neill, D. Fagan, E. Galvan, A. Brabazon, and S. McGarraghy. An analysis of Genotype-Phenotype Maps in Grammatical Evolution. In *EuroGP 2010*, 2010.

[21] Michael O'Neill and Conor Ryan. *Grammatical Evolution: Evolutionary Automatic Programming in a Arbitrary Language*, volume 4 of *Genetic programming*. Kluwer Academic Publishers, 2003.

[22] Judea Pearl. *Heuristics: intelligent search strategies for computer problem solving*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1984.

[23] Elaine Rich. *Artificial intelligence*. McGraw-Hill, Inc., New York, NY, USA, 1983.