Research Repository UCD

Title	Dynamic environments can speed up evolution with genetic programming
Authors(s)	O'Neill, Michael, Nicolau, Miguel, Brabazon, Anthony
Publication date	2011
Publication information	O'Neill, Michael, Miguel Nicolau, and Anthony Brabazon. Dynamic Environments Can Speed up Evolution with Genetic Programming. University College Dublin. School of Computer Science and Informatics, 2011.
Series	UCD CSI Technical Reports, UCD-CSI-2011-03
Publisher	University College Dublin. School of Computer Science and Informatics
Item record/more information	http://hdl.handle.net/10197/3571

Downloaded 2023-04-07T10:52:29Z

The UCD community has made this article openly available. Please share how this access benefits you. Your story matters! (@ucd_oa)

© Some rights reserved. For more information

Dynamic environments can speed up evolution with Genetic Programming

Technical Report UCD-CSI-2011-03

Michael O'Neill, Miguel Nicolau, Anthony Brabazon

Natural Computing Research & Applications Group
Complex & Adaptive Systems Lab
School of Computer Science & Informatics
University College Dublin, Ireland
m.oneill@ucd.ie, miguel.nicolau@ucd.ie, anthony.brabazon@ucd.ie

Abstract

We present a study of dynamic environments with genetic programming to ascertain if a dynamic environment can speed up evolution when compared to an equivalent static environment. We present an analysis of the types of dynamic variation which can occur with a variable-length representation such as adopted in genetic programming identifying modular varying, structural varying and incremental varying goals. An empirical investigation comparing these three types of varying goals on dynamic symbolic regression benchmarks reveals an advantage for goals which vary in terms of increasing structural complexity. This provides evidence to support the added difficulty variable length representations incur due to their requirement to search structural and parametric space concurrently, and how directing search through varying structural goals with increasing complexity can speed up search with genetic programming.

1 Introduction

The application of genetic programming (GP) to dynamic environments is an open issue in the field, which has received increasing attention in recent years [11, 2] and has received greater attention in the broader evolutionary computation community (e.g., see [1, 9, 12]). A recent study by life scientists [6], which used

genetic algorithms to simulate evolutionary processes, discovered that operating in a dynamic environment can facilitate and speed up evolutionary search when adopting a more traditional GA-like representation on boolean landscapes. Speed ups were particularly evident in environments which contained varying goals.

In this study we examine whether or not varying goals (i.e., a dynamic environment) can speed up evolution with genetic programming. Given the variable-length nature of a GP representation our algorithms must be capable of searching both structural and parametric (content) space of a solution concurrently. To study varying goals with GP we must, therefore, consider that goals can vary in both a structural and parametric manner. This is a significant distinguishing novelty of this study over earlier research [6] which focused on fixed-length representations and did not consider the impact of structural variation of goals.

Following our analysis of the types of varying goals which can occur with GP, we present an empirical investigation which questions whether or not it is possible to achieve a speed up in evolution in a dynamic environment compared to an equivalent static environment. We also directly compare the relative importance of varying goals in terms of structural variation and parametric variation.

The paper continues in Section 2 with some background on earlier research in this domain, before presenting our analysis of the types of varying goals which occur in GP in Section 3. We then define the experimental setup adopted in Section 4. After results are presented in Section 5 we draw conclusions and suggest avenues for future research in Section 6.

2 Background

Dynamic environments exist in many real-world problems, and present significant challenges to automatic model induction methods such as genetic programming. An environment can be dynamic in different ways. For example, the frequency of environmental change is one variable. Change could occur so frequently that it takes place within the lifetime of any one individual in a population, it could take place randomly, slowly, in cycles etcetera. Another variable is the degree of change, that is, when a change occurs what is the magnitude of the change. Again, multiple variations might be possible. When considering change in a problem, changes might occur at different levels, for example to the constraints, parameters, and/or functionality of the problem domain.

A number of strategies can be employed by search algorithms operating in a dynamic environment, and we will give a sample of these here. Three excellent books cover the majority of issues in this area and should be read by the interested reader [1, 9, 2]. In order to adapt evolutionary algorithms to dynamic environments researchers have employed a range of strategies. These include, the adoption of evolvable representations which facilitate adaptation. Strategies to maintain population dispersion and diversity in an effort to stem premature

convergence, with examples including the adoption of multiple subpopulations. Mechanisms are adopted to exploit memory either through the implicit memory embedded in the evolving population, or for example, through the explicit storage of individuals which have been successful in the past. Finally, problem decomposition can be used to facilitate adaptation of individuals to changing environmental conditions [1, 9, 2].

3 Varying goal types in GP

When one thinks of dynamic environments in the context of problem solving search algorithms, one tends to consider these types of environments as posing an additional challenge above and beyond static problems. A recent study by life scientists has turned this idea on its head by demonstrating that, under certain circumstances, dynamic environments can actually help evolutionary search find solutions more rapidly than in an equivalent static environment [6]. In this paper we wish to examine if these findings translate to evolutionary computation and in particular GP.

When facing a dynamic (or even static) problem with GP one faces the additional burden of having to concurrently search both structural and parametric space over fixed length representations such as the classic genetic algorithm (GA) [5, 3]. A search of structural space, and therefore a variable-length representation, is necessary in GP as we do not know the size or topology of the solution when setting out to tackle a problem [7]. A parametric search is conducted in GP concurrently to structural search, as to succeed the algorithm must “optimise” the parameters of the evolving GP trees or instructions. For example, the optimal value or content of each node in a GP tree must be found for the “optimal” structure.

The earlier research by Kashtan et al. [6] on the speed up varying environments can bring, conducted simulations using a “GA-like”, fixed-length representation. To extend this research to GP we therefore need to consider how goals can also vary in terms of structure to account for the variable-length representation of GP.

Kashtan et al. [6] found that speed ups in evolution are most likely to occur for a special type of varying goal, termed *modular varying goals* (MVG’s). A modular varying goal is defined such that

“...each new goal shares some of the subproblems with the previous goal...”. [6]

Making this more specific, it is best to examine what the authors mean by sharing subproblems. For example, they use the boolean function

$$(a \text{ XOR } b) \text{ OR } (c \text{ XOR } d)$$

stating that it could be described as a composition of three functions

$$f(g(a,b), h(c,d)),$$

where $h(c, d) = c \text{ XOR } d$, $g(a, b) = a \text{ XOR } b$ and the function f combines its arguments using boolean *OR*. Examples of modular variants of this function are:

$$(a \text{ XOR } b) \text{ AND } (c \text{ XOR } d)$$

and

$$(a \text{ XOR } b) \text{ OR } (c \text{ OR } d).$$

Note that the structure of the function is not altered under the modular varying goals considered earlier by Kashtan et al. [6]. This is not surprising as fixed-length representations were adopted in this earlier study so it was not necessary to investigate goals which vary structurally. In the context of GP, we must consider the search of the structural space of solutions, so it is possible to define a new type of modular varying goal which changes in terms of structural modules. In this study we extend the notion of a modular varying goal to include the idea of a *structural varying goal* (SVG).

We define an SVG in terms of the Kashtan et al. [6] definition of an MVG. Therefore, we define an SVG such that

“Each new goal shares some of the subproblems with the previous goal, with the new goal containing a different number of subproblems from the previous goal.”.

Following from the earlier example, structural varying goals of the primary goal, $(a \text{ XOR } b) \text{ OR } (c \text{ XOR } d)$, might include:

$$(c \text{ XOR } d)$$

and

$$(a \text{ XOR } b) \text{ OR } (c \text{ XOR } d) \text{ OR } (e \text{ XOR } f).$$

4 Experimental Setup

In this study we conduct an empirical investigation to 1) ascertain if dynamic environments can speed up evolution with GP, and 2) to determine the relative importance of the structural versus modular varying goals.

In terms of structural varying goals (Question 2) we consider two setups. In the first setup a change in structure occurs without any consideration for the degree (size) of structural change. In the second structural varying goals setup we restrict structural change to occur in such a way that the complexity of structure increases over time. Earlier research in developmental evaluation (e.g., [4]) has found that evaluating the fitness of a GP individual by combining the performance of intermediate developmental forms of the resulting mature “adult” form of the GP individual on increasing complex versions of the goal during the development of a GP solution can improve the quality (performance and

generalisation) of solutions found. It is interesting to determine if in a dynamic environment across phylogenetic (evolutionary) timescales a dynamic environment with increasing complexity can also play an important role in evolution versus the earlier studies over ontogenetic (developmental) timescales.

The investigations are conducted using two dynamic benchmark symbolic regression problems.

Figure 1: A comparison of the structural varying goals of the Symbolic Regression Problem A to the primary target in the range adopted.

Figure 2: A comparison of the primary target function and its modular variants on Problem A in the range adopted.

4.1 Problem A: $x + x^2 + x^3 + x^4 + x^5 + x^6 + x^7 + x^8$

In this dynamic benchmark the primary target function is

$$x + x^2 + x^3 + x^4 + x^5 + x^6 + x^7 + x^8$$

with 20 input-output test cases drawn from the range -1 to 1. Fitness is simply the sum of the squared error.

The performance of four setups are compared. The first setup, denoted **Static** is the baseline static version of the problem where the target does not change over evolutionary time. The second setup (labelled **MVG**) adopts modular varying goals, such that at each environmental change a goal is selected randomly from the following functions:

Target: $x + x^2 + x^3 + x^4 + x^5 + x^6 + x^7 + x^8$

MVG1: $x + x^2 - x^3 + x^4 + x^5 + x^6 + x^7 + x^8$,

MVG2: $x + x^2 + x^3 + x^4 + x^5 + x^6 - x^7 + x^8$,

MVG3: $x + x^2 + x^3 - x^4 + x^5 + x^6 + x^7 + x^8$,

MVG4: $x + x^2 + x^3 + x^4 + x^5 - x^6 + x^7 + x^8$

The third setup (**SVG**) examines structural variants of the target such that upon each environmental change a goal is selected randomly from the functions:

Target: $x + x^2 + x^3 + x^4 + x^5 + x^6 + x^7 + x^8$

SVG1: $x + x^2 + x^3 + x^4$ and

SVG2: $x + x^2 + x^3 + x^4 + x^5 + x^6$.

For the fourth setup (INC) we examine a special case of the SVG setup where the complexity of the structural variants increase over evolutionary time. The targets are identical to those for the SVG setup (namely, Target, SVG1 and SVG2) however for the first 30 generations SVG1 is the goal, for the following 20 generations SVG2 is the goal, followed by SVG1 for the following 20 generations. The environment cycles between SVG2 and SVG1 until for the final 20 generations (generations 130 to 150) the goal is the actual Target ($x + x^2 + x^3 + x^4 + x^5 + x^6 + x^7 + x^8$). For the SVG and MVG setups the final 20 generations of a run also share the same goal (i.e., the primary target of this problem).

The primary target function and the structural variants are plotted in the range investigated in Fig. 1. The modular variants on Problem A are plotted against the primary target in Fig. 2.

4.2 Problem B: $0.3 * x * \sin(2 * \pi * x)$

A second dynamic symbolic regression benchmark instance is examined where the primary target is

$$0.3 * x * \sin(2 * \pi * x).$$

As per Problem A, 20 input-output test cases drawn from the range -1 to 1 are used with fitness calculated as the sum of the squared error. Again, four setups are compared.

Figure 3: A comparison of the structural varying goals of the Symbolic Regression Problem B to the primary target in the range adopted.

Figure 4: A comparison of the primary target function and its modular variants on Problem B in the range adopted.

In the **Static** setup the goal over the entire evolutionary time examined is the primary target function $0.3 * x * \sin(2 * \pi * x)$.

The functions randomly selected from at each environmental change under the MVG setup are drawn from:

Target: $0.3 * x * \sin(2 * \pi * x)$
MVG1: $0.3 * x + \sin(2 * \pi * x)$,
MVG2: $0.3 + x * \sin(2 * \pi * x)$,
MVG3: $0.3 * x * \sin(2 + \pi * x)$,
MVG4: $0.3 * x * \sin(2 * \pi + x)$

For the **SVG** setup, goals are randomly selected from:

Target: $0.3 * x * \sin(2 * \pi * x)$
SVG1: $\sin(2 * \pi * x)$,
SVG2: $x * \sin(2 * \pi * x)$

As per Problem A, the fourth setup (**INC**) changes goals moving from SVG1 to SVG2, then the primary target and back to SVG1, SVG2 etcetera before settling on the primary target function for the final 20 generations. In setups **MVG** and **SVG** the final 20 generations also use the primary target ($0.3 * x * \sin(2 * \pi * x)$) as the goal.

The primary target function and the structural variants are plotted in the range investigated in Fig. 3. The modular variants on Problem B are plotted against the primary target in Fig. 4.

4.3 GP Parameter Settings

In this study a grammar-based form of genetic programming [8], Grammatical Evolution [10, 2], is employed. We use a simple grammar to generate prefix expressions as detailed in Fig. 5 below. The function set contains standard arithmetic operators, including protected division. Note that for Problem B ($0.3 * x * \sin(2 * \pi * x)$) the function set does not include **sin** or any other trigonometric operator, and the terminal set includes neither π or the constants 0.3 or 2. This presents additional challenges to finding a solution to this target, and suitable structures must be uncovered to endode each of this missing terminals.

```

<expr> ::= (<op>(<expr>, <expr>)) | <var>
<op>   ::= + | - | * | %
<var>  ::= x

```

Figure 5: The grammar adopted.

The evolutionary parameters adopted were, a population size of 100 running for 150 generations. A generational algorithm with elitism of 25% of the total population size is employed with tournament selection with a tournament size of 10% of the population. The probability of standard GE ripple crossover is 0.9, and integer mutation is 0.01. A ramped-half-and-half approach to initialisation is adopted with a maximum derivation tree initialisation depth limit of 15. Wrapping is not employed.

Figure 6: Average best fitness for each environmental setup on Problem A.

5 Results

We now present the results of experiments on both dynamic benchmark problems. Recall that we wish to 1) ascertain if dynamic environments can speed up evolution with GP, and 2) to determine the relative importance of the structural versus modular varying goals.

Examining plots of the average best fitness over 100 runs in Figs. 6 and 7 we observe differences in behaviour between setups adopting varying goals and the static environment. Note that in these Figures in the last 20 generations (130-150) the target function is always the primary target in each case. We measure relative success of the different types of dynamic environments by assessing the average best fitness over these final generations.

When examining the fitness profiles in Figs. 6 and 7 note that not every setup is being applied to the primary target function ($x+x^2+x^3+x^4+x^5+x^6+x^7+x^8$ or $0.3*x*\sin(2*\pi*x)$) at any point in time (except for the final 20 generations as noted above).

For the modular varying goal setup (MVG) we can see fitness improve over the first 30 generations, with a corresponding fall off in average best fitness performance when the target function changes at generations 30, 50, 70, 90, 110 and 130. Following each environmental change the population makes performance gains before the next ensuing environmental change takes hold. During the last

Figure 7: Average best fitness for each environmental setup on Problem B.

twenty generations it is worth noting that the MVG setup does not outperform the static benchmark.

Examining performance of the structural varying goal environments namely, **SVG** and **INC** we see the later outperform the static benchmark over the final twenty generations, while the performance of **SVG** is similar to **Static**.

Analysing the average size (see Figs. 8 and 9) of the evolving solutions under each setup, where size is the number of expressed codons used to generate a symbolic expression we can see that the **INC** setup has the lowest genome lengths on both problems. This observation provides some evidence to support the more structurally controlled approach to structural search achieved in the **INC** setup which adopts goals of increasing structural complexity.

It is worth highlighting the fact on both problems the most significant difference in performance is observed when the structural varying goals are incremental in structural complexity, starting from structurally simpler variants of the primary target to increasingly more complex structures.

The results therefore suggest that 1) it is possible to achieve speed up in evolution with GP when a dynamic environment is employed, and 2) the most likely form of varying goal to lead to a speed up with GP is where the goals are structurally varying in a manner such that their complexity is increasing over evolutionary time.

Figure 8: Average expressed lengths of genomes for each environmental setup on Problem A.

6 Conclusions & Future Work

We presented a study on the utility of varying goals with genetic programming for symbolic regression to determine if they might be able to result in a speed up in evolution when compared to a static environment. Given the variable-length nature of a genetic programming representation our algorithms must be capable of searching both structural and parametric space of a solution. As such we explicitly considered *modular varying goals* (MVG's) and in addition introduced the concept of *structural varying goals* (SVG's). We then empirically investigated whether varying goals can speed up GP evolution, and analysed the relative importance of MVG's, SVG's and a special case of SVG's where the goals increase in structural complexity over evolutionary time.

These results suggest that when applying genetic programming to a symbolic regression problem that adopting a dynamic environment which adopts varying goals can speed up evolution and improve the quality of solutions found. In addition, the most notable performance gains occur when structural varying goals exist such that the varying structures are of increasing complexity over evolutionary time.

Overall it is worth noting that the adoption of a seemingly more challenging dynamic environment with genetic programming can lead to improved perfor-

Figure 9: Average expressed lengths of genomes for each environmental setup on Problem B.

mance both in terms of the quality and speed with which solutions are found.

Future research will be focused on determining if these findings generalise to other problem domains, and if alternative approaches to MVG's and SVG's can further improve the performance of evolutionary search. From a practitioners standpoint it is also important to highlight the potential limitations of the approach adopted in this study. When one is dealing with a *real-world* problem of symbolic regression we may have no idea of the structural form of the underlying model. As such to adopt varying goals in practice, and in particular to ensure those goals are structurally varying, would require innovative solutions to be developed. One potentially fruitful approach might be to apply functional transformations to the output values for each given input, where those transformations are designed to be equivalent to structural variations. To successfully achieve this one would need to find correlations between structural variations and functional (i.e., phenotypic or behavioural) variations. The key finding of this study is that assuming the existence of structurally varying goals a speed up in evolution may be possible, and as such future research in this domain is warranted.

Acknowledgments

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

References

- [1] J. Branke. *Evolutionary optimization in dynamic environments*. Springer, 2001.
- [2] I. Dempsey, M. O'Neill, and A. Brabazon. *Foundations in Grammatical Evolution for Dynamic Environments*. Studies in Computational Intelligence. Springer, 2009.
- [3] D. E. Goldberg. *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley, 1989.
- [4] T.-H. Hoang, D. Essam, R. I. McKay, and X. H. Nguyen. Developmental evaluation in genetic programming: A TAG-based framework. In T. L. Pham, H. K. Le, and X. H. Nguyen, editors, *Proceedings of the Third Asian-Pacific workshop on Genetic Programming*, pages 86–97, Military Technical Academy, Hanoi, VietNam, 2006.
- [5] J. H. Holland. *Adaptation in Natural and Artificial Systems*. MIT Press, 1975.
- [6] N. Kashtan, E. Noor, and U. Alon. Varying environments can speed up evolution. *Proceedings of the National Academy of Sciences*, 104(34):13711–13716, 2007.
- [7] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.
- [8] R. I. McKay, X. H. Nguyen, P. Whigham, Y. Shan, and M. O'Neill. Grammar-based genetic programming: a survey. *Genetic Programming and Evolvable Machines*, 11(3-4):365–296, 2010.
- [9] R. W. Morrison. *Designing Evolutionary Algorithms for Dynamic Environments*. Springer, 2004.
- [10] M. O'Neill and C. Ryan. *Grammatical Evolution: Evolutionary Automatic Programming in a Arbitrary Language*. Genetic programming. Kluwer Academic Publishers, 2003.
- [11] M. O'Neill, L. Vanneschi, S. Gustafson, and W. Banzhaf. Open issues in genetic programming. *Genetic Programming and Evolvable Machines*, 11(3-4):339–363, 2010.
- [12] S. Yang, Y.-S. Ong, and Y. J. (Editors). Special issue on evolutionary computation in dynamic and uncertain environments. *Genetic Programming and Evolvable Machines*, 7(4), 2006.