Dynamic Environments Can Speed Up Evolution with Genetic Programming

Michael O'Neill
Natural Computing Research & Applications Group
Complex & Adaptive Systems Laboratory
School of Computer Science & Informatics
University College Dublin
m.oneill@ucd.ie

Miguel Nicolau
Natural Computing Research & Applications Group
Complex & Adaptive Systems Laboratory
School of Computer Science & Informatics
University College Dublin
miguel.nicolau@ucd.ie

Anthony Brabazon
Natural Computing Research & Applications Group
Complex & Adaptive Systems Laboratory
School of Business
University College Dublin
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.

Categories and Subject Descriptors

I [**Computing Methodologies**]; I.2 [**Artificial Intelligence**]; I.2.2 [**Automatic Programming**]

General Terms

Algorithms, Experimentation

Keywords

dynamic environments, genetic programming, modular varying goals, structural varying goals

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, 8, 12]). A recent study by life scientists [5], 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 [5] which focused on fixed-length representations and did not consider the impact of structural variation of goals.

2. 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 [1, 8, 2]. 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 [5]. 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) [4, 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 [6]. 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. [5] 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-

GECCO'11, July 12–16, 2011, Dublin, Ireland.
ACM 978-1-4503-0690-4/11/07.

length representation of GP. Kashtan et al. [5] 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...". [5]

To account for the structural search space of GP we define a structural varying goal (SVG) in terms of the Kashtan et al. [5] 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.".

3. EXPERIMENTAL SETUP

In this study a grammar-based form of genetic programming [7], Grammatical Evolution [10, 2], is employed. 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. The investigations are conducted using two dynamic benchmark symbolic regression problems with the primary target in each case being $x+x^2+x^3+x^4+x^5+x^6+x^7+x^8$ and $0.3*x*sin(2*\pi*x)$. Further details on the experimental setup are available [9].

4. RESULTS

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 the results we observe differences in behaviour between setups adopting varying goals and the static environment. 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 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 an manner such that their complexity is increasing over evolutionary time.

A more detailed exposition of the results is available [9].

5. 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. It was observed that the adoption of a seemingly more challenging dynamic environment with genetic programming can lead to improved performance both in terms of the quality and speed with which solutions are found.

Acknowledgments

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

6. 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] J. H. Holland. *Adaptation in Natural and Artificial Systems*. MIT Press, 1975.

[5] 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.

[6] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.

[7] 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.

[8] R. W. Morrison. *Designing Evolutionary Algorithms for Dynamic Environments*. Springer, 2004.

[9] M. O'Neill, M. Nicolau, and A. Brabazon. Dynamic environments can speed up evolution with genetic programming. Technical Report UCD-CSI-2011-03, School of Computer Science & Informatics, University College Dublin, `http://www.csi.ucd.ie/biblio`, 2011.

[10] M. O'Neill and C. Ryan. *Grammatical Evolution: Evolutionary Automatic Programming in a Arbitrary Language*. 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.