Noname manuscript No.
(will be inserted by the editor)

Emergence in genetic programming

Let's exploit it!

Anikó Ekárt

Received: date / Accepted: date

Abstract Banzhaf explores the concept of emergence and how and where it happens in genetic programming [1]. Here we consider the question: what shall we do with it? We argue that given our ultimate goal to produce genetic programming systems that solve new and difficult problems, we should take advantage of emergence to get closer to this goal.

Keywords emergence · self-modification · autoconstructive evolution · multilevel genetic programming

1 Introduction

The study of emergence is clearly important in many research fields, only in the last five years over one million scientific articles on the topic have been written (based on Google Scholar).[1]

Given the many subtle variants and examples in the literature people may become confused. As opposed to Holland refusing to give a concise definition for the notion of emergence [4], Banzhaf goes back to the origin of the term and adopts the simple definition by which emergence is the process that leads to the whole being more than a sum of its parts [1]. For the genetic programming community, this definition offers clarity and allows the practitioners to move forward toward understanding how emergence happens in general and what it can mean for genetic programming systems.

A. Ekárt
Aston University
Aston Triangle, Birmingham B4 7ET, United Kingdom
E-mail: a.ekart@aston.ac.uk

[1] In comparison, about 17,300 articles have been written on genetic programming in the same period.

Banzhaf [1] discusses the two main mechanisms that would allow for emergence: top-down and bottom-up causation and places them in the context of genetic programming. Selection exemplifies top-down causation, while genotype-phenotype mapping and phenotype to fitness mapping exemplify bottom-up causation. There are plenty of well-documented and analyzed examples in the genetic programming literature to support the existence of emergent phenomena, such as bloat, evolvability, cooperation, modularity and repetitive patterns. He argues that repetitive patterns emerge "as a result of the presence of downward causation realized through selection".

2 Engineering emergence in genetic programming

Genetic programming systems have always been engineered to produce emergent solutions. Through the use of genetic programming, we expect the computer to solve a problem without explicitly telling it what steps to take and in what order, often not knowing ourselves in advance what the solution should be. Then it should not come as a surprise that some emerging properties, such as bloat in many cases, may be unwanted and hinder the production of acceptable solutions, so mechanisms to discourage them have to be put in place. External observation of these unwanted properties and external intervention to reduce or more radically eliminate them is needed.

An interesting idea is internal emergence that allows a system to take advantage of the emergent patterns without external intervention. To achieve this, self-modification and self-reflection are required [6].

3 Exploiting emergence

Substantial effort in the genetic programming community is dedicated to exploring avenues for designing solutions to hard problems. In an ideal situation the solution would adapt to unseen new problems in a satisfactory manner. In our view, this can be achieved to a great extent if *mechanisms for allowing perpetual emergence and autonomous response to emerging properties*, not requiring external intervention, are put in place. Although some authors are of the opinion that genetic programming may not be suitable for perpetual emergence due to the representation and artificial selection mechanism used [2], we believe that the extension of genetic programming systems in the direction of self-modification can overcome the difficulties and allow for more open-ended emergence and exploitation of emerging properties. We shall briefly discuss three promising directions.

Self-modifying Cartesian genetic programming In addition to the usual functions, in self-modifying Cartesian genetic programming the genotype includes primitive functions that act on the genotype itself, allowing the phenotype to unfold over time [3]. Self-modifying Cartesian genetic programming is relatively easy to implement, has been reported successful on a variety of problems

and also has the nice property that self-modification is only triggered when needed and without external intervention.

Autoconstructive evolution Spector proposes to extend genetic programming through so-called autoconstructive evolution, where the reproduction and variation algorithms are encoded in the programs themselves [5]. In this way, these algorithms can evolve together with the problem-solving part of the solutions. The usual encoded restrictions on these mechanisms are removed, so more open-ended emergence is allowed.

Multilevel genetic programming Multilevel selection has been designed to encourage cooperation between multiple partial solutions to solve complex problems [7]. The particularly attractive reported side effect of multilevel selection is the autonomous decomposition of the tackled problem through evolution, without human intervention.

4 Conclusion

We agree with Banzhaf's key idea [1] that selection in genetic programming is an example of top-down causation, while genotype-phenotype mapping and phenotype-fitness mapping exemplify bottom-up causation. The challenge is whether we can engineer genetic programming algorithms to exploit emergence to produce good solutions to real life problems. To this end, we are encouraging increased efforts in the direction of incorporating mechanisms for perpetual emergence and autonomous response to emergent properties.

References

1. W. Banzhaf, Genetic programming and emergence, Genetic Programming and Evolvable Machines, (2013)
2. A.D. Channon and R.I. Damper, Towards the evolutionary emergence of increasingly complex advantageous behaviours, International Journal of Systems Science, 31(7), 843-860 (2000)
3. S. Harding. W. Banzhaf and J. F. Miller, A survey of self modifying Cartesian genetic programming, In R. Riolo et al. (Eds.), Genetic Programming Theory and Practice VIII, p. 91-107 (2011)
4. J. Holland, Emergence, Philosophica, 59, 11- 40 (1997)
5. L. Spector, Towards practical autoconstructive evolution: self-evolution of problem-solving genetic programming systems, In R. Riolo et al. (Eds.), Genetic Programming Theory and Practice VIII, p. 17-33 (2011)
6. S. Stepney, Programming unconventional computers: dynamics, development, self-reference, Entropy, 14, 1939-1952 (2012)
7. S. Wu and W. Banzhaf, Rethinking multilevel selection in genetic programming. In N. Krasnogor et al. (Eds.), Proceedings of the International Conference on Genetic and Evolutionary Computation (GECCO-2011), p. 1403-1410 (New York, 2011)