Genetic Programming and Evolvable Machines (2023) 24:21
https://doi.org/10.1007/s10710-023-09469-9

COMMENTARY

Is the evolution metaphor still necessary or even useful for genetic programming?

Jason H. Moore[1]

Accepted: 13 October 2023 / Published online: 22 November 2023

I read with great interest the historical review of Koza's first book (a.k.a. "Jaws") on genetic programming (GP) by Langdon. There is no question that this book has had a big impact on research and practice in artificial intelligence (AI), artificial life, and machine learning. It certainly inspired my own lifelong work on developing and applying GP algorithms and software when I first read it as a graduate student at the University of Michigan in the 1990s. In fact, one of my first activities as a new graduate and assistant professor was to implement Koza's symbolic regression code in LISP and adapt it to do symbolic discriminant analysis with application to biomedical data. The evolution metaphor used by GP, and its predecessor the genetic algorithm, was very appealing to me as a student because it was easy to understand, and my biology training taught me that evolution by natural selection is a powerful tinkerer and problem solver in nature. Now that I have nearly 30 years of experience with GP, I can't help but question whether the evolution metaphor is still necessary or even useful?

One of the foundational concepts that stood out for me when I was learning about GP is representing solutions as computer programs. This notion is appealing because it gives the user tremendous flexibility in thinking about how to solve a problem. As Langdon [1] notes, Koza used expression trees as a convenient data structure for representing solutions. Others have used linear- or graph-based representations. This flexibility has allowed developers to use GP to generate art, music, electrical circuits, game-playing algorithms, mathematical functions, object designs, and biological models, for example. Perhaps one of the most interesting GP representation examples is the use of Backus-Naur form (BNF) grammars in grammatical evolution. Here, a computer program is generated by applying the rules of the BNF

Thirtieth Anniversary of Genetic Programming: On the Programming of Computers by Means of Natural Selection.

This comment refers to the article available online at https://doi.org/10.1007/s10710-023-09467-x.

✉ Jason H. Moore
Jason.Moore@csmc.edu

[1] Cedars-Sinai Medical Center, Los Angeles, USA

grammar to a string of symbols. This really speaks to the essence of GP as BNF grammars are used to define the syntax of programming languages.

As one thinks about an ideal computational representation for solving a problem it is important to keep in mind several factors. First, is the representation amenable to search algorithms? Second, is the execution of the candidate computer programs feasible given available resources? Third, has the representation been designed to allow domain-specific knowledge to be incorporated? Fourth, is the representation amenable to interpretation? Fifth, does the representation allow the solution to be easily deployed or adopted? These questions reflect the importance of the solution representation in constructing an effective GP. Once an effective and efficient representation has been created it is time to think about search and optimization. This is where the evolution metaphor has historically played a role.

Central to the evolution metaphor is the concept of selection or survival of the fittest. A typical GP will start by generating hundreds or thousands of solutions randomly. Each one is evaluated for its ability to solve the problem and then a subset is selected for further consideration. New solutions can only come from further random generation and/or introducing variation into those that have been selected. This is typically done by exchanging pieces of different solutions or by random changes to solutions. This is often referred to as crossover and mutation consistent with the evolution metaphor. Numerous methods for selection and variation have been developed.

There are certainly advantages of using the evolution metaphor for GP as it provides an intuitive explanation for how the algorithm works and it provides inspiration for the genetic operators used for introducing variation. However, it is also true that GP and other evolutionary algorithms have received criticism from theoretical computer scientists because it lacks strong mathematical theories, and the time complexity can be prohibitive. This has historically cast some shadows on evolutionary computing causing some to steer toward more mathematically rigorous or simpler algorithms even if they might be less effective.

But is the evolution metaphor necessary? Deep learning has had success in domains such as image analysis and is also based on the nature-inspired metaphor of neural networks. However, as deep learning has matured, it can be argued that the neuron metaphor has been lost in favor of a general recognition that it is just a large network of simple mathematical functions. Indeed, deep learning is used as the phrase to refer to this class of algorithms without the neural network reference. It is also true that theory has not kept pace with the many new methods for implementing deep learning. This seems less important in the era of data science where the focus is more on solving real-world problems. Does it matter that we don't have a theory for how a deep learning algorithm classifies medical images if it performs well and we can pinpoint the parts of the images the information is coming from?

As with deep learning, maybe it is time to ween ourselves off the evolution metaphor and present GP as a powerful engine for building computer programs that solve complex real-world problems. Emphasizing the program representation along with a variety of selection and variation operators that are not tied to natural processes could propel GP into the mainstream and help motivate a new generation of computational and data scientists to add this methodology to their toolbox along with

deep learning and large language models. Trying to force various elements of the algorithm, such as representation and variation operators, to be more like the ones found in nature places unnecessary constraints on their design and makes it difficult for researchers not well versed with biological concepts to contribute to their development. Perhaps a name change is in order too. As a computational biologist with education in biology, genetics, and evolution, it is difficult to imagine rebranding and refocusing the algorithm that Koza made famous in the first Jaws. However, our overarching goal as scientists is to help people and help society. Maybe a new GP paradigm will help achieve this goal as deep learning has done.

Funding Open access funding provided by SCELC, Statewide California Electronic Library Consortium.

Reference

1. W.B. Langdon, Jaws 30. Genet. Program. Evolvable Mach. (2023). https://doi.org/10.1007/s10710-023-09467-x