Explainable A.I.: the promise of Genetic Programming Multi-run Subtree Encapsulation

Daniel Howard
Mark A. Edwards
Howard Science Ltd, Malvern, UK
dr.daniel.howard@gmail.com

***Abstract*— Deep Learning and other Artificial Neural Network based solutions are rarely transparent, and white-box solutions are often called for. This paper explains how Multi-run Subtree Encapsulation can provide equivalent white box solutions to facilitate Explainable Artificial Intelligence.**

Keywords*— *Explainable Artificial Intelligence, A.I., Genetic Programming, Evolutionary Computation, modularization, Subtree Encapsulation, Automatically Defined Functions, Software Evolution, white box, black box, expression simplification, Deep Learning, Artificial Neural Networks, Multi-run Subtree Encapsulation, subtree database.

I. Multi-run Subtree Encapsulation

Howard Science software achieves an equivalent white-box solution to an existing black-box or neuromorphic solution such as by Deep Learning to a commercial or industrial problem. Although performance against test data may validate the black-box model, for some requirements, this validation is insufficient – there is a need to understand how inputs combine to get to outputs and why. There is a need for Explainable Artificial Intelligence.

This paper describes a capability of Howard Science based on an algorithm first introduced in 2001 by the author of the present paper, D. Howard, and his then co-authors S. C. Roberts and J. R. Koza [1-3]. Known as Multi-Run Subtree Encapsulation (MRSE) this atomizes encapsulated subtrees (ESTs) as terminals (building blocks) across runs. MRSE is a less well-known strategy of modularization in Genetic Programming (GP) [4] than Automatically Defined Functions (ADFs) [5]. While ADFs can be thought of as subroutines, ESTs are function subprograms.

MRSE is the first successful EST scheme [2-3] since earlier efforts at ESTs did not demonstrate clear advantages over standard GP. It was not until the multi-run idea was paired with ESTs that real benefit was discovered for problems such as parity problems which standard GP could not solve but that both MRSE GP and ADFs GP could indeed solve efficiently [2].

While, GP experiments in [4-5] adopt the practice of taking random test points such that GP delivers a robust result, an important note about MRSE is that it only holds if the test points for the GP fitness function cannot change during the MRSE runs. This is not *generally* allowed as the equivalence between subtrees relies on the constancy of test points. For many problems of practical interest, however, this restriction is inconsequential.

II. Development of MRSE: original motivation

Development of MRSE followed from an idea to use ESTs to save compute time in evaluating evolving GP trees. Crossover is important in standard GP. Figure 1 illustrates a crossover operation. Two GP individuals (left of figure row above) participate in crossover with two possible offspring (left of figure row below). If the result of evaluating the subtrees below the cross over points, over all of the problem test points (could be thousands of points in Computer Vision [1]) are stored in a database (right of figure) as soon as these subtrees should appear in the population, then the evaluation of the offspring takes fewer floating point operations because the pre-computed values are available from storage.

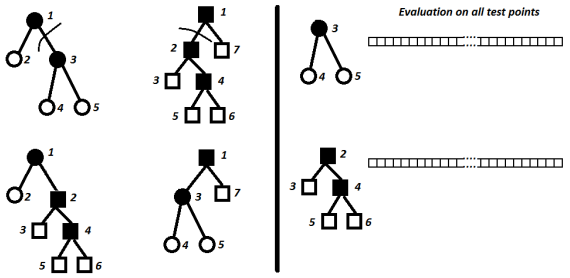

Figure 1 Crossover subtree database calculation savings.

In practice, however, this scheme does not guarantee faster computation because it depends on how big the database is and how long it takes to search it and to retrieve stored evaluations for the subtrees. Several variables that depend on problem; coding; OS and hardware render the use of the database to assist with computation advantageous or not as compared to recalculating the subtrees for the new offspring.

Another use of the subtree database that was posited at that time gave rise to MRSE: that is to encapsulate/atomize and use ESTs across contiguous GP runs/computations. For simplicity, the following describes an MRSE with a multi-run hierarchy of two runs:

- Run GP on the problem for 5 generations.
- Select several ESTs and atomize them (name them as new GP terminals).
- Add these to the original GP terminals to initialize another brand-new run of GP.
- Rely on a terminal mutation operator during this second GP run to ensure this large number of new terminals makes its presence felt in many GP trees as they evolve. Run GP to completion.

978-1-7281-0404-1/19/$31.00 ©2019 IEEE
DOI 10.1109/iCMLDE.2018.00037

III. Encapsulated Subtrees and MRSE

The original motivation for using EST had been to store partial computations into a database during a GP run such that subtrees could be obtained from memory on the fly and not having to be re-computed when a new individual created by a genetic operation, typically crossover involved computing the subtree. As part of that database registration process, subtrees of different sizes appeared that had the same exact (or extremely close) evaluation vector as shown in Figure 2, see [1].

As far as the computational problem is concerned subtrees of different makeup but with the same evaluation vector are the same subtree! This is important.

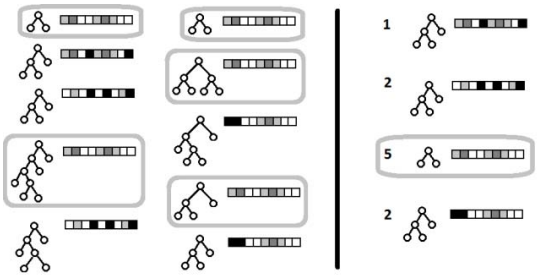

Figure 2 Subtree Database illustration.

Figure 2 provides more detail by means of an illustration. The presence of subtrees in all GP trees in the run is on the left. Subtrees that evaluate to the same result vector are assumed to be the same subtree. If we were to try to algebraically simplify these subtrees there is a good chance that this would be so, but they are equal or almost equal only at the test points.

On the right of Figure 2 is a count for each subtree type. Frequency is a sign of a common building block. There is ambiguity as to which subtrees constitute useful material, useful building blocks. This is important as MRSE must select a number of these ESTs as terminals for the next run. Should one select the most frequent subtrees? Should one select those who emanate from GP trees of high frequency. How should the ESTs for atomization for the next run of MRSE be selected? Strategies that have proved successful have chosen these randomly or by their frequency [1-3]. In figure 2 one subtree type has a count (frequency) of 5 and is circled.

Arguably, the success of MRSE at solving the parity problem [2-3] is owed to the mobility and diversity that is a property of such encapsulations and atomization into GP terminals. The secret to solving the parity problem is to discover the XOR building block and so this atomization assists greatly in this. Standard GP is inefficient at solving the parity problems, indeed it usually fails while GPs that use ADFs or MRSE do solve the parity problems [2].

Another advantage of MRSE over standard GP is compute- time. MRSE stops each run only after 5 or so generations One of the fundamental problems of GP is bloat but this becomes a problem as the code gets very big. EST as terminals pack a punch as they embody significant algebraic expressions decreasing the need for large GP trees. Arriving at the answer is quicker, evolution is quicker than with standard GP [1].

IV. MRSE's role in Explainable A. I.

Notice in Figure 2, that the simplest subtree structure is selected to represent the subtree class. Thus, when the solution is obtained that contains ESTs these are computationally simple, and, in principle, the resulting overall expression can be computed efficiently: the evaluation is organized for the EST terminals can be computed once on the data and applied wherever they should appear (as terminals) and also the ESTs themselves are if not the simplest, the simplest of the GP discovered algebraic expressions.

This has the potential to reduce computational time but more importantly for Explainable A.I. it also eases the understandability of the resulting expressions. Either by direct use of GP or by reverse engineering a neuromorphic solution, any process that eases simplification of the resulting GP expressions is of great benefit to Howard Science as it sheds light on the workings of black-box approaches that are based on Artificial Neural Networks such as Deep Learning.

References

[1] Roberts S.C., Howard D., Koza J.R. (2001), "Evolving modules in Genetic Programming by subtree encapsulation". In Julian F. Miller and Marco Tomassini and Pier Luca Lanzi and Conor Ryan and Andrea G. B. Tettamanzi and William B. Langdon editors, Genetic Programming, Proceedings of EuroGP'2001, volume 2038, pages 160-175, Lake Como, Italy, 2001. Springer-Verlag.

[2] Roberts S.C., Howard D., Koza J.R. (2001), "Subtree encapsulation versus ADFs in GP for parity problems". In Lee Spector and Erik D. Goodman and Annie Wu and W. B. Langdon and Hans-Michael Voigt and Mitsuo Gen and Sandip Sen and Marco Dorigo and Shahram Pezeshk and Max H. Garzon and Edmund Burke editors, Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001), page 186, San Francisco, California, USA.

[3] Daniel Howard (2003). "Modularization by Multi-Run Frequency Driven Subtree Encapsulation". In Rick L. Riolo and Bill Worzel editors, Genetic Programming Theory and Practice, chapter 10, pages 155-171. Kluwer, 2003.

[4] Koza J. (1992), Genetic Programming: On the Programming of Computers by Means of Natural Selection: v. 1 (Complex Adaptive Systems), MIT Press, Cambridge.

[5] Koza J. (1994), "Genetic Programming II: Automatic Discovery of Reusable Subprograms", MIT Press.