Genet Program Evolvable Mach (2017) 18:369–372
DOI 10.1007/s10710-017-9291-2

COMMENTARY

Genotype–phenotype mapping implications for genetic programming representation: Commentary on "On the mapping of genotype to phenotype in evolutionary algorithms" by Peter A. Whigham, Grant Dick, and James Maclaurin

Anikó Ekárt[1] · Peter R. Lewis[1]

Published online: 24 February 2017

Abstract Here we comment on the article "On the mapping of genotype to phenotype in evolutionary algorithms," by Peter A. Whigham, Grant Dick, and James Maclaurin. The article reasons about analogies from molecular biology to evolutionary algorithms and discusses conditions for biological adaptations in the context of grammatical evolution, which provide a useful perspective to GP practitioners. However, the connection of the listed implications for GP is not sufficiently convincing for the reader . Therefore this commentary will (1) examine the proposed principles one by one, challenging the authors to provide more supporting evidence where felt that this was needed, and (2) propose a methodical way to GP practitioners to apply these principles when designing GP representations.

Keywords Genotype–phenotype mapping · Representation · Practical guidelines for GP representation design

1 General comments

It is well-known that evolutionary computation can produce near-optimal solutions to a broad range of complex problems. This discussion paper is based around the assumption that the evolutionary search process, since it has been "sucessful" in nature, must itself be an optimal or near-optimal process. This, the authors argue,

This comment refers to the article available at doi:10.1007/s10710-017-9288-x.

Anikó Ekárt
a.ekart@aston.ac.uk

Peter R. Lewis
p.lewis@aston.ac.uk

[1] Aston Lab for Intelligent Collectives Engineering (ALICE), Aston University, Birmingham, UK

has been used to justify much work aiming at making evolutionary algorithms more biologically plausible, with the expectation to result in improved performance.

In general, we find the idea of optimality of solutions implying an optimality of process intriguing. Indeed, major scientific discoveries have often resulted from completely unusual processes that may have seemed at first sight unproductive and unlikely to lead to any breakthrough. In the article it is argued that since we know that evolutionary processes (usually) work, or work given sufficient time, this then implies (near-)optimality of the evolutionary process itself. We would like *to read further justification for this apparent implication*. To substantiate, how this is indeed an assumption made by many researchers in evolutionary computation and drives much of evolutionary computation work, we would like *to see evidence and relevant references*.

With this said, we agree with the authors that it is very important for evolutionary computation researchers and advocates to be clear about the role of the biological inspiration and evolutionary metaphor in metaheuristic search techniques, and where and how it can help or hinder. Every opportunity should be taken to clarify any potential misconceptions.

The paper's main supporting pillars are a subset of four out of the nine characteristics proposed by Sterelny for robust evolutionary replicators:

C4 separation between genotype and phenotype for optimality;
C6 stable and predictable mapping, where redundancy in behaviour via multiple expressions is possible;
C8 smooth mapping (essentially locality); and
C9 modularity.

While a distinction between genotype and phenotype is not strictly necessary in an (artificial) evolutionary system (standard GP providing one example), C4 suggests a split can provide such a separation of concerns that aids effectiveness. Characteristics C6–C8, although not necessarily arrived at from a biological viewpoint, but an empirical computational experimentation perspective, are well known and, in our opinion, feature in many, though not all, effective applications of evolutionary computation.

Interestingly, the authors chose grammatical evolution (GE) for illustration. In its aim to provide a generic representation, GE has inherent properties that violate Sterelny's characteristics of effective genotype–phenotype mappings. Therefore the effectiveness of using both generic representations and generic operators at the same time is being questioned, despite their apparent success in nature. When a generic representation is used, the human effort shifts to the design of the genotype–phenotype mapping rather than the design of the representation or operators. In this case, the implication is that the operators have to be designed with the mapping in mind. Since one of the typical benefits of employing a generic representation is to be able to employ generic operators, is this benefit lost? What then is the implication for the design of artificial evolutionary systems in practice? Is the quest to design generic approaches to automated problem solving doomed to failure?

2 From principles to practical guidelines

Motivated by this discussion, the authors propose five principles for genetic programming representation design. We agree that representation is one crucial aspect for the success or otherwise of an evolutionary (and in fact computational intelligence) solution, but not the only one. Looking at the first four principles at high level, they can be seen as also emerging from evolutionary computation or more generally from computer science, without necessarily being strongly rooted in biology:

1. Maintaining building blocks: in GP, these could be emerging as automatically defined functions or built into the representation (such as more complex mathematical functions in symbolic regression).

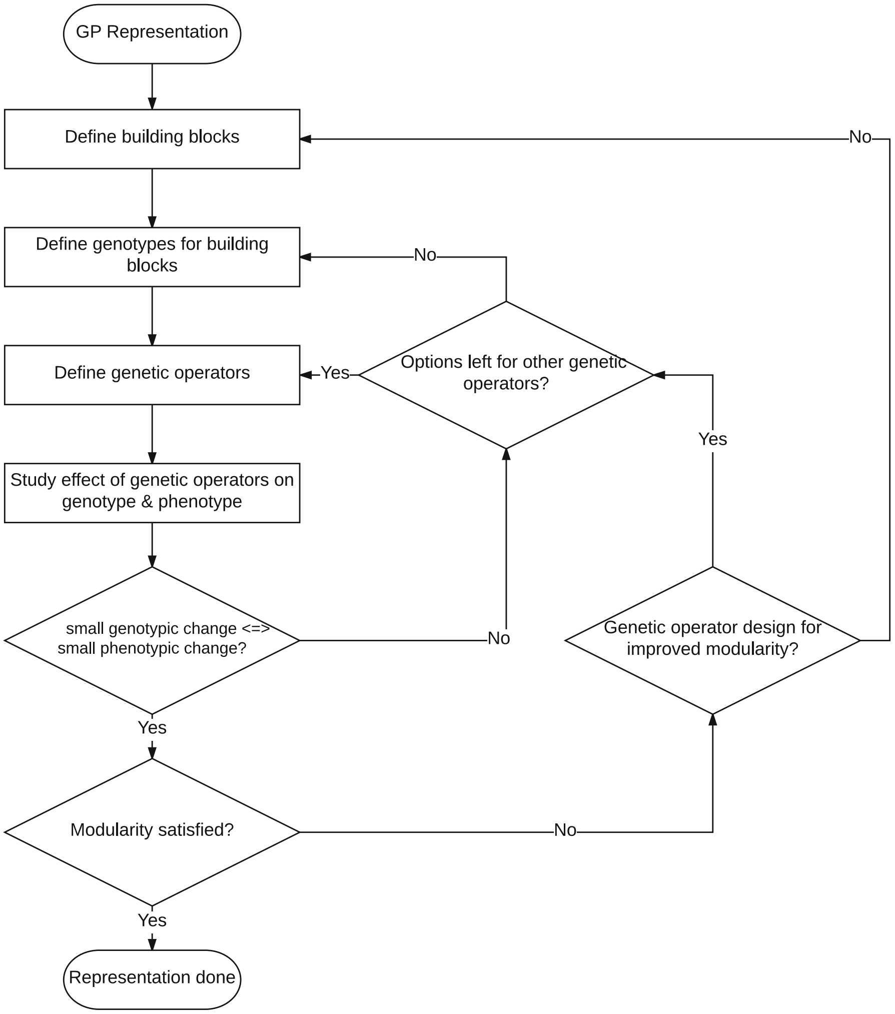
GP Representation
Define building blocks
Define genotypes for building blocks
Define genetic operators
Study effect of genetic operators on genotype & phenotype
small genotypic change <=> small phenotypic change?
Modularity satisfied?
Representation done
Options left for other genetic operators?
Genetic operator design for improved modularity?
Yes
No

Fig. 1 Steps to design a GP representation according to presented principles

2. Although somewhat confusing in its statement (when using representation and behaviour, the terms search space and problem space become unclear), this principle advocates the design of genetic operators consciously of their effects in the phenotype space. It is not quite clear how the practitioner, once aware of the said relationship, should take it into account in the design of the genetic operators. We would like *to see a concrete example.*
3. Stability under small change follows on from the previous principle and is a somewhat controversial principle here. For GP, locality can be used as a measure of difficulty. Also, in addition to the authors' example, in a generic binary genetic algorithm a small change in the genotype can have rather different effect on the phenotype when the genotpye is mapped onto the phenotype, yet this genetic algorithm itself can be very effective on some problems, with no restrictions on operators necessary. We would like *to see an example of how this would be achieved in practice, without limiting evolution such that an acceptable solution become beyond reach.*
4. Modularity has been advocated in programming since the early days.

It is actually rather nice to see how there could be biological justification to this research community's empirical findings. Our final point is that if the authors wanted to convince the reader to strictly follow Sterelny's characteristics (and here we are neither advocating nor questioning the correctness of the approach) when developing a GP solution, we argue that a more structured set of guidelines or steps would be needed. We endeavour to propose the processes illustrated in the flowchart in Fig. 1. The principles are present in the processes in the flowchart. We added what we feel are supportive decision points that allow the practitioner to go back to appropriate previous steps and start again, should the (partial) representation solution fail the expectations.