When and Why Development Is Needed

Generative and Developmental Systems

Alexandre Devert*
TAO/LRI
CNRS, Univ Paris-Sud
F-91405, France
devert@mica.edu.vn

ABSTRACT

Within the evolutionary computation community, there is a strong consensus to agreed on the need of indirect representations to achieve scalability. But no such consensus has been yet found on how to design an indirect representation. An idea to build a scalable representation, is to see the phenotype to genotype mapping as an iterative transformation process: an explicit development stage. But such an approach is computationally expensive and then it relevance might be questionable. Through a simple, accessible example, optimization of a block stack overhang, it is shown that, indeed, an explicit development stage can be the only way if one wants a scalable representation and/or scalable solutions to a problem.

Categories and Subject Descriptors

I.2.8 [**Problem Solving, Control Methods, and Search**]: Heuristic methods

General Terms

Algorithms, Design

Keywords

Evolutionary Algorithms, Generative and Developmental Systems

1. INTRODUCTION

One of the strength of the evolutionary algorithms (EAs) resides in the few hypothesis they require to solve a problem. That strength comes at a price : compared to more informed approaches (gradient based optimization, linear programming, etc.), evolutionary algorithms usually require more fitness evaluations. A worrying problem with EAs is the scalability of the representation: usually, the larger is the phenotype, the larger is the genotype, making a lot of problems quickly intractable. This issue have been and is still problematic when targeting large phenotypes like building structures, integrated circuits or neural networks.

Nonetheless, some successful achievements in the evolution of large and complex phenotypes have been done, like [2] and [3, 4] with respectively large neural networks and metallic trusses. One fruitful idea behind those achievements is the usage of *indirect representations*. With indirect representations, the genotype is no longer an explicit representation of an object. An object is implicitly represented as the result of an interpretation process, the genotype to phenotype mapping. L-Systems and Cellular Automatas are famous examples of such implicit definitions: a few rules can generate very complex objects. Thus very compact genotypes can code for very large phenotypes, eventually making large design accessible with conventional EAs.

The mathematics, computer graphics or biologic modelization fields provide us plenty of ways to implicitly defines objects and structures. The enumeration of those representations is beyond the scope of this work (See [1, 7] for a survey of representations). We will rather focus on a distinction within the possible genotype to phenotype mappings : those which features an explicit development stage, and those which not. When using explicit development, the genotype to phenotype mapping takes place as following. Either hand-made or defined in the genotype, we have an initial phenotype. A function, defined in the genotype, is iterated over the phenotype, modifying it at each step. At some point, the iteration are stopped and the final phenotype is the one which is used for the evaluation step. Such a definition of explicit development includes L-Systems and Cellular Automatas.

The iterations computed during an explicit development stage can make a potentially very computationally expensive genotype to phenotype mapping. Indeed, Gauci et al. in [2] showed an effective alternative to explicit development. In their work, the genotype is a function, and the phenotype is the output generated by this function, from an elementary, hand-made input. The genotype function is constructed as a composition of elementary functions, chosen to express symmetries and repetition when composed. Gauci et al. claim that this approach features an *implicit development*, because the phenotypes obtained by their approach share traits (repetition, symmetries) with what is usually obtained with explicit development, without the need of expensive iterations. Indeed, is there situations where an explicit development stage is indispensable ? Is there always a way to make development *implicit* ?

2. BLOCK STACKING

We introduce here a simple example : maximization of the overhang of a balanced block stack. Block stacks are two dimensions physical constructions made of identical blocks of

*Currently at MICA, Ha Noi University of Technology, (CNRS/UMI2954, Grenoble IMP) Viet Nam

GECCO'09, July 8–12, 2009, Montréal Québec, Canada.
ACM 978-1-60558-325-9/09/07.

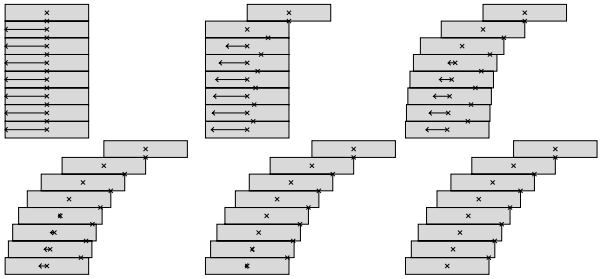

Figure 1: Iterative representation of an harmonic stack. A transformation, when iterated, turns any stack into the harmonic stack

length 1, stacked on each other. Balanced block stacks only are considered: those that do not collapse. If we consider only stacks where at most one block can be stacked over a given block, then the optimal overhang is known and is unique : it is achieved by the *harmonic stacks* [6].

The harmonic stack is usually defined recursively. To build the *harmonic stack* H_{N+1} with $N+1$ block, we should drop one block at the horizontal position 0, then we stack over it the harmonic stack H_N made of N blocks, shifted by $\frac{1}{2N}$. Here it is an alternative definition of H_N, an iterative one. Take any stack of N blocks. Then, shift each block independently and simultaneously of $\left|G_i - \frac{1}{2}\right|$, where G_i is the gravity center of the blocks 1 to $i-1$ (top block is block 1). By iterating this transformation, the stack will converge to H_N. The recursive definition "hide" the importance of the gravity centers, while the iterative definition clearly shows it.

Such definitions can be found with the right intuition and some maths, the problem is quite simple. But what happens if we consider, for a time, that the maximization of the overhang of a block stack is a hard problem, for which we want to use an EA to get a solution ? Encoding individually the position of each blocks is bound to fail. The more blocks we have, the more variables to optimize we have, thus a direct approach can not be scalable. An indirect representation is needed, to have a genotype independent from the number of blocks. An explicit development stage would work, because it fits the iterative definition of an harmonic stack. The genotype would be a function $F(G(i))$, represented as an universal function approximator (a multi-layer perceptron). The phenotype would consists at shifting the block i by $F(G(i))$, and repeating that until a fixed point or a maximal number of iterations is reached. This genotype size is independent from the number of blocks, and once the optimal function is found, it works for any numbers of blocks. Thus, such an explicit development stage would provides a scalable representation and scalable solutions.

Is it possible to obtain the same advantages with an implicit development stage ? Without explicit iterations, the positions of the blocks have to be specified directly from their index, by a function for instance. It fits the recursive definition of an harmonic stack, where we specify a block position directly from it index. So does evolving the function that returns a block position from it index is a sound approach ? Since no center of gravity information is exploited, with such a representation, higher number of blocks would lead to poor results. High numbers of blocks require very fine tuning of the blocks positions to maintain balance. So even very little alterations of the genotype are likely to break the balance of a given block stack. Thus the scalability would be broken. When computing the positions of the blocks in one shot, the center of gravity information is useless, it is exploitable only in an iterative fashion, ie, the iterative representation of an harmonic stack. The recursive definition is a kind of shortcut, giving the right answer without giving the path to it.

3. CONCLUSION

When designing an indirect representation to solve a problem with EAs, one might be concerned by scalability issues. One design decision is should we rely or not on an explicit development phase for the genotype to phenotype mapping. Such a phase can make the mapping computationally very expensive. On one example, maximum overhang of stacked blocks, it is shown that indeed, the iterative way is the only way to have a scalable representation. The key is that the iterative approach is able to exploit an essential information : the centers of gravity. Such an information allow to represent high stacks without adding difficulties compared to stacks with low numbers of blocks. Thus, at least in that case, explicit development is unavoidable. Moreover, experiments have been done. The same EA (CMA-ES) have been used by the author to optimize the overhang of stacks with increasing number of blocks, by encoding functions as universal approximators (multi-layer perceptrons). The explicit development approach was performing perfectly, independently from the number of blocks. Two implicit development approach suffered from the balance adjusting problem as predicted, and thus where not scalable at all. The shortcut used by an implicit developement approach is equivalent to an imperative definition of an object, whereas explicit development is equivalent to a constructive definition. Imperative definitions are closer to "needles in the haystack", making them hard to be found by a stochastic search.

4. REFERENCES

[1] P. J. Bentley and S. Kumar. Three ways to grow designs: A comparison of embryogenies for an evolutionary design problem. In W. Banzhaf, editor, *GECCO*, pages 35–43. Morgan Kaufmann, 1999.

[2] J. Gauci and K. Stanley. Generating large-scale neural networks through discovering geometric regularities. In Lipson [5], pages 997–1004.

[3] R. Kicinger, T. Arciszewski, and K. A. De Jong. Morphogenesis and structural design: cellular automata representations of steel structures in tall buildings. In *Proceedings of the Congress on Evolutionary Computation (CEC'2004)*, pages 411–418, Piscataway, NJ, 6 2004. IEEE Press.

[4] T. Kowaliw, P. Grogono, and N. Kharma. Environment as a spatial constraint on the growth of structural form. In Lipson [5], pages 1037–1044.

[5] H. Lipson, editor. *Genetic and Evolutionary Computation Conference, GECCO 2007, Proceedings, London, England, UK, July 7-11, 2007*. ACM, 2007.

[6] M. Paterson and U. Zwick. Overhang. In *SODA'06: Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 231–240. ACM Press, 2006.

[7] K. O. Stanley and R. Miikkulainen. A taxonomy for artificial embryogeny. *Artif. Life*, 9(2):93–130, 2003.