EVOLUTIONARY DESIGN USING DEVELOPMENT

Michal Bidlo
Information Technology, 2nd year, full-time study
Supervisor: Lukáš Sekanina

Faculty of Information Technology, Brno University of Technology
Božetěchova 2, 612 66 Brno, Czech republic

bidlom@fit.vutbr.cz

Abstract. The first part of this paper introduces a new classification of developmental systems used in the evolutionary design: (1) the infinite development and (2) the finite development. Two sample applicaions are presented demonstrating the ability of this approach to discover novel solutions in the area of the design of digital circuits. In the second part, a formal model of a general developmental system is proposed and the impact of environment on the complexity of the developing objects is investigated.

Keywords. Evolutionary design, development, environment, program, cellular automaton, digital circuit, rewriting system, string complexity.

1 Introduction

In the recent years, biology-inspired design methods were utilized to solve many complex problems in various engineering and scientific areas. The main advantages of these methods are: (1) the designer designs the evolutionary algorithm, problem encoding and the evaluation function and the evolutionary design system executes autonomously in order to find the target solution and (2) the evolutionary algorithm may discover totally new and innovative solutions which are out of the scope of the conventional design techniques. The areas, in which the bio-inspired techniques have succeeded, include mathematics, physics, mechanical engineering, electronics, computer science and others.

This paper deals with the the artificial systems, whose operation is inspired by the biological principles of *phylogeny, ontogeny* and *embryogeny*, for the design of digital circuits. Phylogeny concerns the temporal *evolution* of a genetic program, the hallmark of which is the evolution of a species. The emergence of living organisms in nature is based upon the reproduction of the program (genotype), subject to an extremely low error rate at the individual level. The *evolutionary algorithm* is a basic computational model inspired by this process [1]. Ontogeny and embryogeny deal with the *development* of multicellular organisms that is based on the successive division of the mother cell, with each newly formed cell possessing a copy of the original genome, followed by a specialization of the daughter cells in accordance with their environment. In the area of computer science and engineering the term *computational development* (or *development* in short) is usually used to cover both the processes of ontogeny and embryogeny. Typical computational models inspired by these principles are *cellular automata* [8] and *Lindenmayer systems* [5]. With respect to the processes by which our artificial design systems are inspired, we call them the *evolutionary developmental systems*.

The objective is to study various approaches to modeling the development in the area of computer science. In Section 2 the common features of the existing developmental models are identified, an

original classification of the developmental systems is proposed and sample applications demonstrating the abilities of the evolutionary design of digital circuits are presented. Section 3 introduces a formal concept of a general developmental system in which the environment can influence the development of the target object and investigates the properties of the proposed approach with respect to the theoretical computer science and computer engineering.

2 Classification of Developmental Systems

The problem of scale is probably the most limiting issue in the evolutionary design. It can be expressed as follows. As the complexity of the target system increases, the length of the chromosome of the evolutionary algorithm increases entailing the search space to become enormously large and the evolutionary algorithm is not able to explore it effectively.

In general, the aim of the development in the evolutionary design is to improve scalability of evolution in order to obtain more complex designs. Moreover, adaptation, reproduction and self-repair are the issues often involved in the artificial developmental process.

Considering the existing models of the development published so far, the following objectives can be identified: (1) the developmental model is utilized in order to overcome the problem of scale and (2) various properties of a specific developmental model are investigated in solving a particular problem. In the first case, we can require the target system to "grow" (develop) for a (theoretically infinite) number of iterations and to be fully functional in the defined iterations (developmental steps). It means that the system can develop continually and infinitely. The second case includes the systems in which we are usually interested in the "final object" that emerged by applying the developmental rules or in the systems in which we do not require the endless development.

According to this classification, two basic approaches to the computational development can be introduced, which represent the first main contribution of this work to the theory of the computational development: (1) the *infinite development* and (2) the *finite development*. Note that the term infinite development relates to the objects being developed rather than to the utilized developmental encoding. This concept will be defined formally. The description of the infinite and finite development is based on the fitness function which determines how the object under development fulfils the requirements specified by the designer.

Definition 1 (infinite and finite development) Let $(e, \Delta) \in \mathcal{C}$ be a chromosome of an evolutionary algorithm, where $e \in \mathcal{S}$ denotes an embryo, $\Delta : \mathcal{S} \rightarrow \mathcal{S}$ denotes a developmental algorithm, $\mathcal{C}$ represents the space of chromosomes and $\mathcal{S}$ represents the space of candidate solutions. Δ is said to have been performing the *infinite development*, if and only if there is *no* finite integer $k > 0$, such that $\Phi(\Delta^d(e)) < F_d$ for all $d > k$, where Φ denotes a fitness function, d denotes the number of applications of Δ and F_d represents a fitness value by which the solution $\Delta^d(e)$ fulfils the requirements specified by the designer. Otherwise, Δ is said to have been performing the *finite development*.

With respect to the results obtained so far in the area of the evolutionary design, we can formulate the following hypothesis related to the evolutionary design of digital circuits using the development.

Hypothesis 1 An evolutionary developmental system utilizing a form of the infinite or finite development is able to discover innovative solutions in the area of the design of digital circuits.

In the following subsections, two sample applications will be presented utilizing the proposed concept of the infinite and finite development. The first model intended to perform the infinite development is based on repeated application of a *program* consisting of simple application-specific instruction for the design of arbitrarily large well-scalable digital circuits. In the case of the finite development a cellular automaton-based model was devised for the design of small combinational

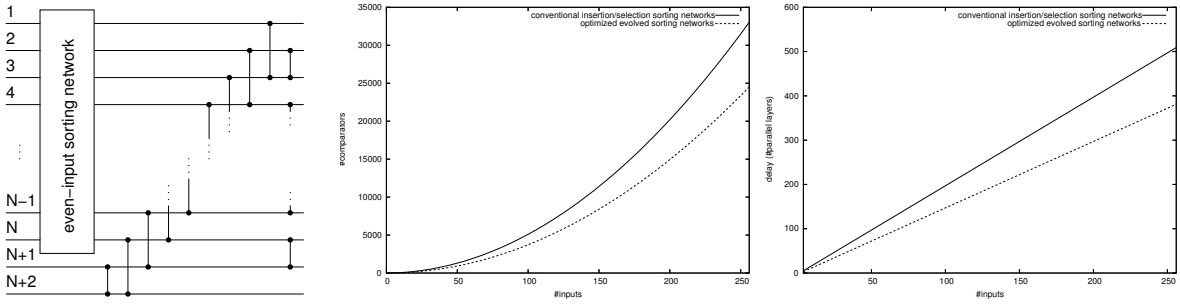
circuits of various classes. The best experimental results will be presented demonstrating the validity of the proposed hypothesis.

2.1 Experiments and Results of the Infinite Development

A special developmental encoding was devised for the evolutionary design of *arbitrarily large* well-scalable combinational circuits at the gate level. Sorting and median networks, adders and parity circuits were successfully designed. Moreover, polymorphic increasing/decreasing sorting networks and odd/even parity circuits were evolved [3].

A trivial instance of the circuit is utilized at the beginning of the developmental process that represents the *embryo*. The developmental encoding utilizes a set of simple application-specific instructions which are intended to manipulate the building blocks of the circuit (copy the gates or modify their input indices). A finite sequence of instructions represents a *program* (constructor) for the development of the embryo. After a single application of the constructor on the embryo, a larger circuit emerges. Then the constructor is applied on the result of the previous application and, again, larger circuit is created and so on. The objective is to design (using the evolutionary algorithm) constructors, which are able to create theoretically infinitely large circuit.

A perfect constructor was evolved for the sorting networks, whose properties are substantially better in comparison with the conventional algorithms of the same class (insertion or selection principle). The developed sorters possess lower number of comparative operations which imply lower production cost, and shorter delay entailing faster sorting (see Figure 1) [7]. This result was classified as human-competitive in the international competition 2005 Human-Competitive Awards which was held together with the Genetic and Evolutionary Computation conference (GECCO 2005) in Washington D.C., US. Moreover, the evolved algorithm was proved to be general [2].



(a) (b) (c)

Figure 1: (a) The principle of the best evolved constructor for even-input sorting networks, (b) the number of comparators, (c) delay of these networks in comparison with the conventional approach.

2.2 Experiments and Results of the Finite Development

In contrast to the infinite development, in some cases we require (usually a single) target object to be developed in a finite number of developmental steps without any interest in its functionality in the next developmental steps. We were interested in the design of digital circuits in this case. For example, 2-bit multipliers, 6-input sorting networks or 14-bit parity circuits were successfully developed.

For the experiment of the finite development, a *generative cellular automaton* has been introduced. This model works in the same way as the ordinary cellular automaton but the difference is that the generative cellular automaton introduces a *symbol* associated with each rule of the local transition function to be generated by the cells which apply these rules in order to determine the next state. For

the purposes of generating digital circuits, the number of cells of the automaton equals the number of primary inputs of the target circuit. The symbols represent logic gates which are generated by each cell during the development of the automaton. The inputs of the gates being generated are connected to the primary inputs of the circuit to be designed in case of the first developmental step, otherwise they are connected to the outputs of the gates generated in the previous developmental step.

Evolutionary algorithm was utilized to search the initial configuration of the generative cellular automaton and the rules of the local transition function together with the gates to be generated. Since uniform cellular automaton has been considered, the local transition function is identical for all the cells. The number of states of the automaton and the number of developmental steps determining the delay of the circuit to be developed were determined experimentally. Figure 2 shows the development of the best evolved 2-bit multiplier possessing delay 2τ (the best currently known 2-bit multiplier possesses delay 3τ [6]).

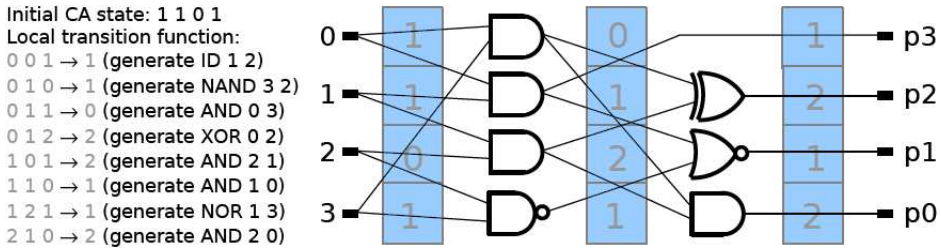


Figure 2: The development of the best evolved 2-bit multiplier using a generative cellular automaton and the finite development. The cellular automaton with the initial configuration (1101) develops according to the local transition function with each cell generating a logic gate as described in the left side of the figure. The expression $u\ v\ w \rightarrow x$ represents the combination of the states of the cells in the defined neighborhood followed by the new state of the middle cell. Note that zero-boundary conditions are applied, i.e. the upper neighbor of the top cell and the lower neighbor of the bottom cell are treated as zero-state cells.

3 Formal Concept of a General Developmental System

Many developmental models have been introduced in the area of computer science. However, only a fraction of them considers an interaction of genotype or phenotype with an environment surrounding the developmental system as usual in nature. Moreover, no formal concept of development has been proposed yet.

We will introduce an original approach to the formal description of a general developmental system which is intended as a basis for investigating various developmental models and their properties. An environment surrounding the developing object and an algorithm determining rigorously the developmental process with respect to the environment will be introduced. The definition constitutes the second main contribution of this work to the theory of the computational development.

Definition 3 (developmental system) A *developmental system* is a foursome $\Delta = (D, E, e, M)$, where D is a rewriting system (called D system – see [4]), E is a finite environment alphabet, $e \in E^*$ is an *environment*, where E^* is the set of all strings over E, and M is a *developmental algorithm*. If $E = \emptyset$, then e, in fact, does not exist and the system does not interact with the environment, otherwise the system is said to have been developing in the environment e.

In fact, the D system can be expressed as a type-0 grammar which is able to represent an arbitrary algorithm-based model. The developmental algorithm can be, for example, represented by a Turing

machine which is a general computational model, i.e. an arbitrary control of the developmental process and the interaction with the environment can be established.

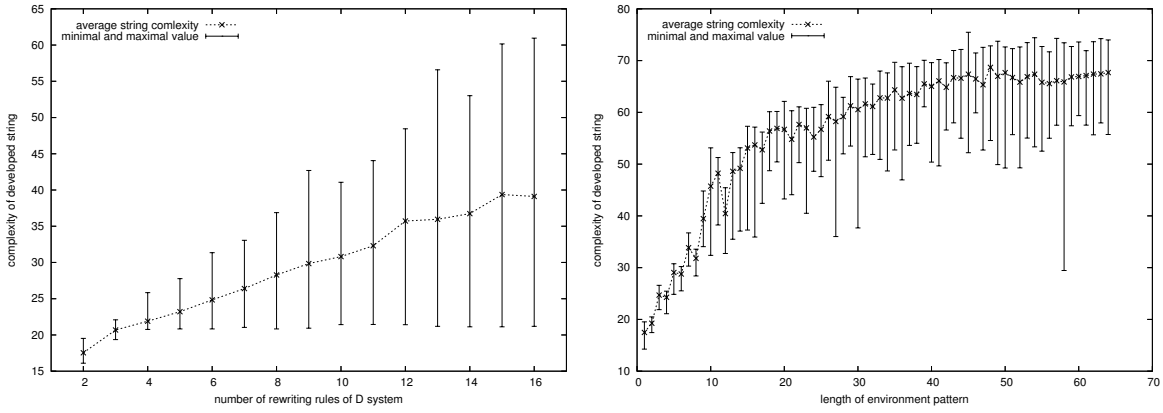
4 The Impact of Environment on the Complexity of Developed Objects

Our experience from nature and the experiments performed so far in this area suggest that the utilization of the environment must lead to generating much more complex structures in comparison with the case when no environment is considered. Therefore, the following hypothesis can be formulated.

Hypothesis 2 If an evolutionary developmental system utilizes a piece of information from the chromosome (i.e. the genetic information) and another piece of information from environment (i.e. an external information), then more complex entities can be evolved in comparison with the situation when no environment is utilized and only the genetic information is considered for the development.

The evolutionary algorithm was utilized to design the rewriting rules of the D system and a finite binary *environment pattern* repeating periodically along the string generated by the D system. The developmental algorithm was determined a priori that controls deterministically the developmental process and involves the environment whose symbol 0, respective 1 forbid, respective permit the application of the rewriting rules at the given positions of the developed string. The objective was to design such developmental system that generates as complex strings as possible from a single starting symbol. The complexity is measured as the number of symbols (bytes) of the compressed string using a LZW-based algorithm.

Experiments were conducted with the developmental system without the environment increasing the number of rewriting rules. Then the evnironment was involved in the developmental process and only two developmental rules were considered. The experimental results are shown in Figure 3 [4].



(a) (b)

Figure 3: The dependence of the developed string complexity on (a) the number of rewriting rules of the D system when no environment is considered and (b) the length of the binary environment pattern when only two rewriting rules are considered.

5 Objectives of the Future Dissertation

On the basis of the previous research, I have decided to focus on the following issues related to the area of the computational development: (1) To classify the developmental systems according to

the summary of the properties of existing developmental models. (2) To create a formal model of a general developmental system. (3) To investigate various properties of developmental systems by means of the formal model, especially the impact of environment on the complexity of developed objects. (4) To investigate the properties and abilities of the formal model of development, especially in the applications of the infinite development. (5) To demonstrate the contribution of the proposed approach on selected problems utilizing the knowledges obtained in the previous objectives.

At this point, items 1, 2 and 3 have been finished. Item 5 has been partially solved and item 4 will be investigated in the near future.

6 Conclusion

As obvious from section 2.1 and 2.2, both the approaches to the computational development (the infinite and the finite development) have shown to be able to construct the circuits that improve some current conventional approaches, which confirms the hypothesis 1. Moreover, the experiments have shown that the environment influences substantially the complexity of the developed objects. Therefore, hypothesis 2 has been confirmed too.

Acknowledgment

The research was performed with the support of the Grant Agency of the Czech Republic in the projects under No. 102/04/0737 *Modern Methods of Digital System Synthesis* and No. 102/05/H050 *Integrated Approach to Education of PhD Students in the Area of Parallel and Distributed Systems*.

References

[1] Bäck, T. , Fogel, D. B., Michalewicz, Z. (eds.): Evolutionary Computation, part 1 and 2, Institute of Physics Press, Bristol UK, 2000.

[2] Bidlo, M., Bidlo, R., Sekanina, L.: Designing a Novel General Sorting Network Constructor Using Articial Evolution, Proc. of the International Conference on Computational Intelligence, 2006 (submitted).

[3] Bidlo, M., Sekanina, L.: Providing Information From the Environment for Growing Electronic Circuits Through Polymorphic Gates, Proc. of Genetic and Evolutionary Computation Conference – Workshops 2005, Association for Computing Machinery, pp. 242–248, 2005.

[4] Bidlo, M., Sekanina, L.: On Impact of Environment on the Complexity of Developed Objects, Genetic Programming and Evolvable Machines, 2007 (submitted).

[5] Lindenmayer, A.: Mathematical Models for Cellular Interaction in Development, parts I and II, Journal of Theoretical Biology, No. 18, pp. 280–315, 1968.

[6] Miller, J. F., Job, D.: Principles in the Evolutionary Design of Digital Circuits – part I, Genetic Programming and Evolvable Machines, Vol. 1, No. 1, pp. 8–35, 2000.

[7] Sekanina, L., Bidlo, M.: Evolutionary Design of Arbitrarily Large Sorting Networks Using Development, Genetic Programming and Evolvable Machines, Vol. 6, No 3, pp. 319–347, 2005.

[8] von Neumann, J.: The Theory of Self-Reproducing Automata, A. W. Burks (ed.), University of Illinois Press, 1966.