Association for Computing Machinery
Advancing Computing as a Science & Profession

Ubiquity, an ACM publication
February, 2013

Ubiquity Symposium

Evolutionary Computation and the Processes of Life

Evolutionary Computation in Physical World

by Lukáš Sekanina

Editor's Introduction

In this ninth symposium article, Lukáš Sekanina addresses evolutionary and evolvable hardware; answering the questions what it means for a physical system to be designed evolutionarily and on what kinds of computations such physical systems perform will be answered.

Editor

Ubiquity Symposium

Evolutionary Computation and the Processes of Life

Evolutionary Computation in Physical World

by Lukáš Sekanina

Evolutionary algorithms (EAs) are population-based search algorithms. They have been applied to solve numerous engineering, as well as scientific design and optimization problems, in many cases outperforming conventional methods or other search-based methods. Dozens of human-competitive results obtained using EAs were awarded at the annual GECCO Humies competitions [1]. Despite these achievements, the EA-based problem solving approach has been criticized mainly because of very long runtimes, insufficient scalability, an inherently stochastic nature, and a missing rigorous theoretical background, especially regarding convergence analysis. In order to speed up the EA, particularly the fitness function evaluation, various accelerators have been proposed utilizing clusters of computers, graphics processing units (GPUs), or specialized hardware.

This article addresses the scenario in which evolutionary design (or optimization) is carried out on a chip and candidate designs are real physical entities such as electronic circuits. It will be shown that this approach exhibits several unique features that are very relevant to the purpose of this symposium. Finally, the question on what it means for a physical system to be designed evolutionarily and on what kinds of computations such physical systems perform will be answered.

Evolvable Hardware

Living organisms show many interesting qualities—adaptation, learning, self-organization, self-repair, self-construction, etc.—we would like to see in our engineering products. They also implement many of their basic functions quite efficiently in comparison to engineering

products. One of the main differences one can observe between living organisms and engineering products is their origin. While living organisms are results of evolution, engineering products are usually designed according to a well-developed methodology and typically in the top-down process. What are the benefits of evolving instead of creating in the context of (physical) hardware?

Physical hardware is evolved in the field of "evolvable hardware." This is possible because reconfigurable chips exist whose physical structure (and function) is defined by a configuration bit stream stored in a configuration memory. Because the configuration bit stream can be modified on the fly, new hardware functionality can be dynamically created. In addition to well-known digital and analog reconfigurable hardware, evolutionary design has also been conducted with more exotic reconfigurable devices such as reconfigurable antennas, reconfigurable optical systems, reconfigurable molecular structures, etc.

In evolvable hardware, electronic circuits encoded as strings of symbols are constructed and optimized by the evolutionary algorithm in order to obtain a circuit implementation satisfying the specification. In order to evaluate a candidate circuit, a reconfigurable circuit (or its simulator if evolution is performed using a circuit simulator)[1] is reconfigured using a new configuration created on the basis of the chromosome (i.e. configuration) content. The configured device is then stimulated by a chosen set of input vectors; its responses are collected and compared with the desired responses. Finally, the fitness score is calculated, which reflects to what extent the candidate circuit satisfies the specification. This process is repeated for every individual generated by EAs. New circuits are created when genetic operators (such as mutation and crossover) are applied on existing circuit configurations. High-scored candidate circuits have a higher probability that their genetic material will be selected for creating future generations.

This is one of the approaches enabling us to construct adaptive hardware capable of reconfiguration and adaptation when new requirements are specified or faults or various fabrication imperfections are present. It was recognized in the embodied intelligence research that a physical body is important for achieving machine intelligence [2]. Evolvable hardware is a step in this direction. Another aim of evolvable hardware is to produce innovative (or even patentable) physical designs. A wire antenna evolved for the NASA ST5 mission is the best-known example [3]. Other examples can be found in domains of analog and digital electrical circuits, quantum circuits, mechanical systems, robots, controllers and optical systems [1, 4].

[1] The approach is then called "extrinsic evolution."

These examples show the search-based design performed by an evolutionary algorithm can find novel designs in the space of possible solutions *S*, which a designer has pre-determined by means of a chosen problem representation and genetic operators. For example, if the objective is to evolve as compact as possible implementation of an *n*-input/*m*-output combinational circuit (implementing function F: $\{0, 1\}^n \rightarrow \{0, 1\}^m$) in a reconfigurable array consisting of *p* programmable 2-input gates, then *S* contains all possible logic networks composed of up to *p* gates, *n* inputs and *m* outputs. It should be noted that *S* comprises all candidate solutions, including those that do not satisfy the specification. The structure of the resulting circuit could be quite unconventional with respect to the state of the art. On the other hand, a typical feature of conventional circuit synthesis and optimization methods is that they never reach some portions of *S* even if there are useful designs.

Unconstrained Evolution

However, if all candidate solutions are physical circuits and they are evaluated directly on a chip (i.e. no circuit simulator is utilized),[2] then the evolutionary design can do some extra work.

Thompson used the FPGA XC6216 chip to evolve a tone discriminator—a circuit discriminating between square waves of 1 kHz and 10 kHz [5]. With 10 x 10 configurable blocks of the XC6216 device, the circuit should output 5V for one of the frequencies and 0V for the other. The problem is an evolved circuit has to discriminate between input periods five orders of magnitude longer than the propagation time of each configurable block. Although the evolved circuit was surprisingly small, Thompson was not able to create a reliable simulation model of it. In addition, he observed the circuit configuration works only with the reconfigurable chip used during evolution; another chip of the same type configured according to the evolved configuration did not produce the desired behavior. He concluded that the evolution was able to explore material characteristics of a particular chip and environment in order to create the required behavior.

Let us suppose that the search space size $|S| = 2^c$ where *c* is the configuration bit stream size. We can see unconstrained evolution provides in some sense more solutions than the extrinsic evolution, where the number of distinguishable solutions is $|S|$ at most. One must recall that introducing more physical aspects (such as temperature, electromagnetic field, material constants, etc.) to the model only partially helps. We never know what evolution will be utilized for building a solution next time.

[2] The approach is called “intrinsic evolution.”

This phenomenon could be problematic for routine circuit design where the goal is to find a configuration solving a given problem and reuse it in different chips. On the other hand, it opens doors to new technological innovations. It was shown that various materials could be configured using EA to perform useful behaviors (e.g. computations) without understanding the underlying physics. The method is known as the "evolution in materio." For example, Harding et al. demonstrated liquid crystals could be configured to perform useful computations [6]. The search-based approach seems to be the only one capable to configure materials without understanding their internal structures.

What Does it Mean for Computation?

First of all, we do not know how and why an evolved solution (e.g. a circuit implementing desired mapping *F*) internally works. The mapping between an abstract computational model, which is always constructed for conventionally designed circuits, and an evolved physical implementation also remains unknown [7]. Although one could inspect the reconfigurable device and build its very precise model, it makes no sense to establish the mapping before the evolution is executed because the evolution could utilize non-modeled features of the reconfigurable device to reach the required behavior. Establishing the mapping at the end of evolution could solve the problem, because it could be possible to identify the physical states corresponding with the abstract computational states. Unfortunately, this is a quite challenging task. Although we could apply all the physical theories we currently know to analyze and explain the evolved system, obtaining a satisfactory answer is not quite certain. According to Bartles et al. [8], the evolution is capable of discovering new physical behaviors that are known to be physically impossible from the point of view of current physical theories.

Let us now consider an evolvable hardware-based robot deployed in space, controlled by a reactive controller implemented using a reconfigurable chip, which is dynamically modified by EAs. The robot interacts with its environment and its mission time is supposed to be endless. During the mission, the reconfigurable chip in fact implements an endless sequence of mappings *F1*, *F2*, *F3*, … each of them potentially non-interpretable. This example can be classified as the super-Turing computation [9, 10, 11].

Conclusions

As engineers we can now construct useful hardware systems using evolution. Evolved systems are uniquely composed and potentially capable of autonomous self-adaptation and self-repair. However, we do not understand them because they were evolved by EAs in real (physical)

environments. EAs can explore the "dark corners" of design spaces which humans have left unexplored.

In the same sense we do not understand living organisms because they are results of natural evolution. Melanie Mitchell argues new developments are needed in theoretical computer science to capture and explain biological computation (i.e. those biological processes that we usually interpret as computation [12]). It is reasonable to believe the same future theories will be useful for better understanding of evolved hardware systems.

Acknowledgments

This work was supported by the Czech Science Foundation project "Natural Computing on Unconventional Platforms" P103/10/1517 and the IT4Innovations Centre of Excellence CZ.1.05/1.1.00/02.0070.

References

[1] Koza, J. Human-competitive results produced by genetic programming. *Genetic Programming and Evolvable Machines* 11, 3-4 (2010), 251-284.

[2] Brooks, R. The relationship between matter and life. *Nature* 409, 6818 (2001), 409-411.

[3] Hornby, G., Globus A., and Linden D. et al. Automated Antenna Design with Evolutionary Algorithms. In *Proc. 2006 AIAA Space Conference* (San Jose, CA, Sept. 19-21). AIAA, Reston, VA, 2006.

[4] Haddow, P. C., and Tyrrell, A. M. Challenges of evolvable hardware: past, present and the path to a promising future. *Genetic Programming and Evolvable Machines* 12, 3 (2011), 183-215.

[5] Thompson, A. Silicon Evolution. In *Proc. of Genetic Programming GP'96* (Stanford, CA, July 28-31). MIT Press, Cambridge, 1996, 444-452.

[6] Harding, S. L., Miller J. F., and Rietman E. A. Evolution in materio: Exploiting the physics of materials for computation. *International Journal of Unconventional Computing* 4, 2 (2008), 155-194.

[7] Sekanina, L. Evolved computing devices and the implementation problem. *Minds and Machines* 17, 3 (2007), 311-329.

[8] Bartels, R. et al. Learning from Learning Algorithms: Applications to attosecond dynamics of high-harmonic generation. *Physical Review A* 70, 1 (2004).

[9] Eberbach, E. Toward a theory of evolutionary computation. *BioSystems* 82, 1 (2005), 1-19.

[10] Sekanina, L. *Evolvable Components: From Theory to Hardware Implementations*. Springer Verlag, Berlin, 2004.

[11] van Leeuwen, J., and Wiedermann, J. The Turing Machine Paradigm in Contemporary Computing. In *Mathematics Unlimited - 2001 and Beyond*. Springer, Berlin, 2001, 1139-1155.

[12] Mitchell, M. Ubiquity symposium: Biological Computation. *ACM Ubiquity* 2011, February.

About the Author

Lukáš Sekanina is a full professor with the Faculty of Information Technology, Brno University of Technology, Czech Republic, where he received all his degrees (Ing. in 1999 and Ph.D. in 2002). He was awarded a Fulbright scholarship to work with NASA Jet Propulsion Laboratory in Pasadena in 2004. He was a visiting lecturer/researcher with Pennsylvania State University, Universidad Politécnica de Madrid, and University of Oslo. Sekanina has served as an associate editor of *IEEE Transactions on Evolutionary Computation* and *Int. Journal of Innovative Computing and Applications*. His research interests include bio-inspired computation and evolvable hardware. See http://www.fit.vutbr.cz/~sekanina

DOI: 10.1145/2435197.2435199

