ABSTRACT
There are different models of evolutionary computations: genetic algorithms, genetic programming, etc. This chapter presents mathematical foundations of evolutionary computation based on the concept of evolutionary automaton. Different classes of evolutionary automata (evolutionary finite automata, evolutionary Turing machines and evolutionary inductive Turing machines) are introduced and studied. It is demonstrated that evolutionary algorithms are more expressive than conventional recursive algorithms, such as Turing machines. Universal evolutionary algorithms and automata are constructed. It is proved that classes of evolutionary finite automata, evolutionary Turing machines and evolutionary inductive Turing machines have universal automata. As in the case of conventional automata and Turing machines, universal evolutionary algorithms and automata provide means to study many important problems in the area of evolutionary computation, such as complexity, completeness, optimality and search decidability of evolutionary algorithms, as well as such natural phenomena as cooperation and competition. Expressiveness and generality of the introduced classes of evolutionary automata are investigated.

INTRODUCTION
In this chapter, we argue that the separation of combinatorial optimization methods into exact and heuristic classes is somewhat superficial. Natural classification of algorithms depends on the complexity of the search problem solved by an algorithm. The, so called, exact methods, can and have to be interrupted to produce approximate solutions for large search problems. As a result, these algorithms become heuristic. This is usually true when somebody tries to use dynamic programming to solve NP-complete problems for big or multidimensional
Evolutionary Computation

systems. As we know, it is possible, for example, to use “exact” dynamic programming for 6-10 cities in traveling salesman problem, but only inexact dynamic programming solutions for hundreds and thousands cities are tractable. Although “inexact” evolutionary algorithms and simulated annealing methods can guarantee finding exact solutions for traveling salesman problem, in a general case, this is possible to do only in an infinite number of generations. However, solving a problem in an infinite number of steps goes beyond classical algorithms and Turing machines, and in spite of being common in mathematics, encounters steady resistance in finitely oriented conventional computer science.

Here we show how to achieve the same results, i.e., to find exact solutions for hard problem, in a finite number of steps (time). Namely, we can use super-recursive algorithms. They allow one to solve many problems undecidable in the realm of recursive algorithms (Burgin, 2005). We argue that it is beneficial for computer science to go beyond recursive algorithms, making possible to look for exact solutions of intractable problems or even to find solutions of undecidable problems, whereas recursive solutions do not exist. As the basic computational model, we take evolutionary automata, which extend computational power of evolutionary Turing machines introduced in (Eberbach, 2005) and parallel evolutionary Turing machines introduced in (Burgin & Eberbach, 2008). Our goal here is to study expressiveness of classes of evolutionary automata, relations between these classes, and existence of universal automata in these classes.

Our chapter is organized as follows. In section 2 we present the relevant background information on evolutionary computation and its theoretical foundations. In section 3 we propose a generic class of evolutionary automata to model evolutionary processes. In particular, evolutionary automata consist of evolutionary finite automata, evolutionary Turing machines and evolutionary inductive Turing machines. The properties of each class are investigated. The corresponding classes of universal evolutionary automata are defined. In section 4 we study the generality of evolutionary automata approach in modeling all known and future subareas of evolutionary computation. In section 5 we make conclusions and outline directions for the future work.

BACKGROUND

Evolution by natural selection is one of the most compelling themes of modern science. In evolutionary algorithms, selection operates on population of individuals represented by semiotic chromosomes, which are stored in a computer’s memory. Populations of semiotic chromosomes evolve in a computational process, using mutation and/or crossover in much the same way as natural populations do. This form of computation is called Evolutionary Computation (EC). Evolutionary Computation consists of four main areas: Genetic Algorithms (GA), Genetic Programming (GP), Evolution Strategies (ES) and Evolutionary Programming (EP). Additional areas include: Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), co-evolution, Artificial Immune Systems (AIS), evolutionary robotics, evolvable hardware, Evolutionary Artificial Neural Networks (EANN), evolutionary multiobjective optimization, Artificial Life (A-Life), Classifier Systems, DNA-Based Computing and some fields in bioinformatics. Applications of Evolutionary Computing are vast and diverse. Evolutionary Computing is used for finding solutions of intractable (hard and NP-complete) optimization problems, machine learning, data mining, neural network training, evolution of technology, robotics, control, electronic circuit design, games, economics, network design, pattern recognition, genome and protein analysis, DNA-based computing, evolvable programming languages, reconfigurable hardware and many others (Back, Fogel and Michalewicz 1997; Fogel 2001; Michalewicz and Fogel, 2004).

However, in spite of a diversity of useful applications, evolutionary computation theory is still very young and incomplete (Fogel, 1995; 2001; Michalewicz, 1996; Michalewicz and Fogel, 2004). Studied theoretical topics include convergence in the limit (elitist selection, Michalewicz’s contractive mapping GAs, (1+1)-ES), convergence rate (Rechenberg’s 1/5 rule), Building Block analysis (Schema Theorems for GA and GP), best variation operators (No Free Lunch Theorem). Very little has been known about expressiveness, or computational power, of Evolutionary Computation and its scalability. Conventional computation has many models. One of the most popular is Turing Machine. Quantum computation has such a theoretical model as Quantum Turing Machine. However until recently, Evolutionary Computation did not have a theoretical model that represented practice in this domain. Of course, there are many results on evolutionary algorithms theory (see, e.g., Holland, 1975;
Related Content

Noise Power Spectrum for Firecrackers
www.igi-global.com/article/noise-power-spectrum-firecrackers/52980?camid=4v1a

Multimodal Genetic Algorithms for Craniofacial Superimposition
www.igi-global.com/chapter/multimodal-genetic-algorithms-craniofacial-superimposition/36313?camid=4v1a

Inferring Gene Regulatory Networks from Genetical Genomics Data
www.igi-global.com/chapter/inferring-gene-regulatory-networks-genetical/38232?camid=4v1a

MP Modelling of Glucose-Insulin Interactions in the Intravenous Glucose Tolerance Test
Vincenzo Manca, Luca Marchetti and Roberto Pagliarini (2014). Natural Computing for Simulation and Knowledge Discovery (pp. 171-183).
www.igi-global.com/chapter/mp-modelling-of-glucose-insulin-interactions-in-the-intravenous-glucose-tolerance-test/80064?camid=4v1a