Editorial

Evolutionary Computation: Theories, Techniques, and Applications

Vincent A. Cicirello

Computer Science, Stockton University, 101 Vera King Farris Dr, Galloway, NJ 08205, USA; cicirelv@stockton.edu

1. Introduction

Evolutionary computation is now nearly 50 years old, originating with the seminal work of John Holland at the University of Michigan in 1975 which introduced the genetic algorithm [1]. Evolutionary computation [2] encompasses a variety of problem-solving methodologies that take inspiration from natural evolutionary and genetic processes. The most well-known form of evolutionary computation is the genetic algorithm [3,4], which evolves a population of solutions to the problem at hand, each represented as a bit-string—the genotype—with a fitness function measuring the fitness of the bit-string within the context of the problem (i.e., mapping a genotype to a phenotype). Evolutionary operators, such as mutation, crossover, and selection, control the simulated evolution over several generations.

There are now many forms of evolutionary computation (a few of which are illustrated in Figure 1) that have developed over the years, including genetic programming [5], evolution strategies [6], differential evolution [7,8], evolutionary programming [9], permutation-based evolutionary algorithms [10], memetic algorithms [11], the estimation of distribution algorithms [12], particle swarm optimization [13], interactive evolutionary algorithms [14], ant colony optimization [15,16], and artificial immune systems [17], among others [18,19]. Among the characteristics of evolutionary algorithms that lead to powerful problem solving is the fact that they lend themselves very well to parallel implementation [20–22], enabling the exploitation of today’s multicore and manycore computer architectures. Rich theoretical foundations also exist which are related to convergence properties [23–25], parameter optimization, and control [26], as well as the powerful analytical tools of fitness landscape analysis [27–29], such as fitness–distance correlation [30] and search landscape calculus [31], among others. These theoretical foundations inform the engineering of evolutionary solutions to specific problems. There are also many open-source libraries and toolkits available for evolutionary computation in a variety of programming languages [32–41], making the application of evolutionary algorithms to new problems and domains particularly easy.

Figure 1. A few of the many forms of evolutionary computation.
Evolutionary computation has been effective in solving problems with a variety of characteristics, and within many application domains, such as multiobjective optimization [42–45], data science [46], machine learning [47–49], classification [50], feature selection [51], neural architecture search [52], neuroevolution [53], bioinformatics [54], scheduling [55], algorithm selection [56], computer vision [57], hardware validation [58], software engineering [59,60], and multi-task optimization [61,62], among many others.

This Special Issue brings together recent advances in the theory and application of evolutionary computation. It includes 13 articles. The authors of the 13 articles represent institutions from 11 different countries, demonstrating the global reach of the topic of evolutionary computation. The published articles span the breadth of evolutionary computation techniques, and cover a variety of applications. The remainder of this Editorial briefly describes the articles included within this Special Issue; and I encourage you to read and explore each.

2. Overview of the Published Articles

This overview of the articles is organized in the order in which the contributions to the Special Issue were published.

Cicirello (contribution 1) presents a new mutation operator for evolutionary algorithms where solutions are represented by permutations. The new mutation operator, cycle mutation, is inspired by cycle crossover. Cycle mutation is designed specifically for assignment and mapping problems (e.g., quadratic assignment, largest common subgraph, etc.) rather than ordering problems like the traveling salesperson. This article includes a fitness landscape analysis exploring the strengths and weaknesses of cycle mutation in terms of permutation features.

Osuna-Enciso and Guevara-Martínez (contribution 2) propose a variation of differential evolution that they call stigmergic differential evolution which can be used for solving continuous optimization problems. Their approach integrates the concept of stigmergy with differential evolution. Stigmergy originated from swarm intelligence, and refers to the indirect communication among members of a swarm that occurs when swarm members manipulate the environment and detect modifications made by others (e.g., the pheromone trail-following behavior of ants, among others).

Córdoba, Gata, and Reina (contribution 3) consider a problem related to energy access in remote, rural areas. Namely, they utilize a (µ + λ)-evolutionary algorithm to optimize the design of mini hydropower plants, using cubic Hermite splines to model the terrain in 3D, rather than the more common 2D simplifications.

Parra, et al. (contribution 4) consider the binary classification problem of predicting obesity. In their experiments, they explore utilizing evolutionary computation in feature selection for binary classifier systems. They consider ten different machine learning classifiers, combined with four feature-selection strategies. Two of the feature-selection strategies considered use the classic bit-string-encoded genetic algorithm.

Fan and Liang (contribution 5) consider directional sensor networks and target coverage. In their approach to target coverage, they developed a hybrid of particle swarm optimization and a genetic algorithm. Their experiments demonstrate that the hybrid approach outperforms both particle swarm optimization and the genetic algorithm alone for the problem of maximizing covered targets and minimizing active sensors.

Wang, et al. (contribution 6) developed a hybrid between particle swarm optimization and differential evolution for real-valued function optimization. Their hybrid combines a self-adaptive form of differential evolution with particle swarm optimization, and they experiment with their approach on a variety of function optimization benchmarks.

Chen, et al. (contribution 7) explore the constrained optimization problem of optimizing the linkage system for vehicle wipers. Their aim was to improve steadiness of wipers. They utilize differential evolution to optimize the maximal magnitude of the angular acceleration of the links in the system subject to a set of constraints. They were able to reduce the maximal magnitude of angular acceleration by 10%.
Tong, Sung, and Wong (contribution 8) analyze the performance of a parameter-free evolutionary algorithm known as pure random orthogonal search. They propose improvements to the algorithm involving local search. They performed experiments on a variety of benchmark function optimization problems with a variety of features (e.g., unimodal vs. multi-modal, convex vs. non-convex, separable vs. non-separable).

Anđelić, et al. (contribution 9) approach the problem of searching for candidates for dark matter particles, so-called weakly interacting massive particles, using symbolic regression via genetic programming. Their approach estimates the interaction locations with high accuracy.

Wu, et al. (contribution 10) developed a recommender system utilizing an interactive evolutionary algorithm for making personalized recommendations. In an interactive evolutionary algorithm, human users are directly involved in evaluating the fitness of members of the population. Wu, et al. use a surrogate model in their approach to reduce the number of evaluations required by users.

Dubey and Louis (contribution 11) utilize a\( (\mu + \lambda) \)-evolutionary algorithm. They developed an approach to deploying a UAV-based ad hoc network to cover an area of interest. UAV motion is controlled by a set of potential fields that are optimized by the\( (\mu + \lambda) \)-evolutionary algorithm using polynomial mutation and simulated binary crossover.

Lazari and Chassiakos (contribution 12) take on the problem of deploying electric vehicle charging stations. They define it as a multi-objective optimization problem with two cost functions: station deployment costs and user travel costs between areas of demand and the station’s location. Their evolutionary algorithm’s chromosome representation combines \( x \) and \( y \) coordinates of candidate charging station locations, using the classic bit-string of genetic algorithms to model whether or not each candidate station is deployed.

Reffad and Alti (contribution 13) use NSGA-II to optimize enterprise resource planning performance. They aimed to optimize average service quality and average energy consumption. They propose an adaptive and dynamic solution within IoT, fog, and cloud environments.

3. Conclusions

This collection of articles spans a variety of forms of evolutionary computation, including genetic algorithms, genetic programming, differential evolution, particle swarm optimization, and evolutionary algorithms more generally, as well as hybrids of multiple forms of evolutionary computation. The evolutionary algorithms represent solutions in several ways, including the common bit-string representation, vectors of reals, and permutations, as well as custom representations. The authors of the articles tackle a very diverse collection of problems of different types and from many application domains. For example, some of the problems considered are discrete optimization problems, while others optimize continuous functions. Although many of the articles focus on optimizing a single objective function, others involve multi-objective optimization. Some of the articles primarily utilize common benchmarking optimization functions and problems, while several others explore a variety of real-world applications, such as optimizing mini hydropower plants, UAV deployment, the deployment of electric vehicle charging stations, target coverage in wireless sensor networks, enterprise resource planning, recommender systems, dark matter detection, and optimizing vehicle wiper linkage systems, among others. The diversity of evolutionary techniques, evolutionary operators, problem features, and applications that are covered within this collection of articles demonstrates the wide reach and applicability of evolutionary computation.

Conflicts of Interest: The author declares no conflicts of interest.
List of Contributions


References


35. Bell, I.H. CEGO: C++11 Evolutionary Global Optimization. *J. Open Source Softw.* 2019, 4, 1147. [CrossRef]


41. Tarkowski, T. Quile: C++ genetic algorithms scientific library. *J. Open Source Softw.* 2023, 8, 4902. [CrossRef]

60. Arcuri, A.; Galeotti, J.P.; Marculescu, B.; Zhang, M. EvoMaster: A Search-Based System Test Generation Tool. *J. Open Source Softw.* 2021, 6, 2153. [CrossRef]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.