Association for Computing Machinery
Advancing Computing as a Science & Profession

Ubiquity, an ACM publication
September, 2012

Ubiquity Symposium

Evolutionary Computation and the Processes of Life

The Essence of Evolutionary Computation

by Xin Yao

Editor's Introduction

In this third article in the ACM Ubiquity symposium on evolutionary computation Xin Yao provides a deeper understanding of evolutionary algorithms in the context of classical computational paradigms. This article discusses some of the most important issues in evolutionary computation. Three major areas are identified. The first is the theoretical foundation of evolutionary computation, especially the computational time complexity analysis. The second is on algorithm design, especially on hybridization, memetic algorithms, algorithm portfolios and ensembles of algorithms. The third is co-evolution, which seems to be under studied in both theory and practice. The primary aim of this article is to stimulate further discussions, rather than to offer any solutions.

Editor

Association for Computing Machinery
Advancing Computing as a Science & Profession

Ubiquity, an ACM publication
September, 2012

Ubiquity Symposium

Evolutionary Computation and the Processes of Life

Essence of Evolutionary Computation

by Xin Yao

Evolutionary computation, as it is studied now, often borrows ideas from biology. However, most of the evolutionary algorithms that have been used in practice are actually much closer to classical artificial intelligence than to biology.

A simple evolutionary algorithm (EA) can often been described as follows:

1. Initialize all individuals in a population at random.
2. Evaluate the fitness of every individual in the population.
3. Select fit individuals probabilistically to form an intermediate parent population.
4. Apply recombination and mutation operators probabilistically to the intermediate parent population, and generate the offspring population.
5. Reproduce the next generation from the combined parent and offspring populations.
6. Terminate the algorithm if the stopping criteria are met. Otherwise, go to Step 2.

The above algorithm is the basis of numerous EAs that have been developed and used for decades. It seems to capture the core idea of natural evolution—survival of the fittest. However, a closer look at the fundamental steps of an EA, it actually shares more similarities to the generate-and-test search algorithm in artificial intelligence than to biological evolution.

The essence of an EA consists of two major steps. One is to **generate** offspring through recombination and mutation. The other is to **select** fit individuals from a population of individuals. In fact, recombination and mutation are merely examples and terms borrowed from biology. We can use any variation operators to generate offspring from parents. This is precisely what the generation step can do in the generate-and-test search algorithm. For selection in an EA, the key objective is to select better individuals in the hope that they would lead to even better individuals in future generations, although this is never guaranteed. In other words, selection is just one way to **test** individual's potential in finding the global optima. There has been a huge amount of research work in evolutionary computation in designing more appropriate variation operators and selection schemes for different problems. The foundation of all these EAs is the generate-and-test search algorithm.

By considering EAs in the classical artificial intelligence domain, one can observe more easily two potential characteristics of EAs. One is the **population**. Virtually all EAs, and all evolutionary computation techniques, rely on populations, rather than a single individual. The only exception is in the theoretical analysis of EAs, where (1+1) EAs were analyzed in the early days. The second characteristic of EAs is **stochasticity**. All EAs are stochastic algorithms. Such stochasticity is essential in ensuring EA's global convergence. In essence, EAs are population-based stochastic generate-and-test search algorithms.

Computational Time Complexity Analysis of Evolutionary Algorithms

Linking EAs to classical artificial intelligence enables us to have a deeper understanding of EAs in comparison with other existing algorithms. For example, in the case of combinatorial optimization, we can use existing theoretical frameworks and techniques to analyze EA's computational time complexity on certain problems. Such analyses help to shed light on the fundamental question of when an EA is expected to perform well on a given problem, and why [1]. More interestingly, we can drill down to more detailed questions such as when a population helps in solving a problem [2, 3], given that population is often regarded as an important characteristic of EAs.

Similarly, analyses can be carried out to understand the impact of mutation and recombination on the performance of EAs [4]. Not only have such analyses been done on artificial problems, they have also been performed on more realistic problems, e.g., the unique input-output sequence problem occurred in finite state machine testing [5], an important area of software testing. Of course, the performance of an EA in solving a problem is not just determined by isolated impact of operators and selection. The interactions among operators and selection can also have a major impact on the performance of an EA. Well- balanced combination of operators and selection can lead to an efficient EA. For example, the balance between mutation rate and selection pressure has been shown to be essential in determining an EA's efficiency in solving some problems [6].

EAs are most often used in practice to solve hard problems, where little domain knowledge is available. In such cases, finding the exact optimal solution is usually infeasible. Instead, approximate solutions are often found. There has been theoretical work since 2002 [7] in analyzing EA's approximation ability in solving difficult combinatorial optimization problems, although the work on this topic has been limited. There have been far more empirical observations than theoretical proofs that EAs are good at finding approximate solutions to hard combinatorial optimization problems.

Although there have been many theoretical results appearing in recent years, they are primarily limited to the domain of evolutionary optimization, especially combinatorial optimization [4]. Few theoretical results are available in evolutionary learning and co-evolutionary learning.

Hybridization, Memetic Algorithms and Algorithm Portfolios

Examining EAs that were reported to be highly effective in solving challenging real-world problems, one observes that most such EAs are hybrid algorithms, e.g., a hybridization between an EA and another algorithm (often a local search algorithm). This is not surprising because local search often makes use of problem-specific heuristics and thus helps to improve EA performance.

However, there is a different perspective toward understanding hybrid algorithms, i.e., search bias [8]. Such a perspective helps to link together several other types of EAs, including memetic algorithms [9], portfolio algorithms [10], algorithm ensembles [11], and other forms of adaptive search operators.

For any given EA, its behavior is determined by its search bias, which can be regarded as a probability distribution (over the entire search space) of visiting points (i.e., individuals) in the entire search space. Ideally, we want to design our EA such that this search bias converges to the global optima quickly. However, we do not know where the global optima are and do not know the problem we are solving. We certainly do not want to use algorithms with inappropriate search bias. What can we do? One approach to deal with this situation is to mix different search biases together in EAs, which can be achieved by hybridization (mixing a classical EA with other heuristics), portfolio algorithms, algorithms ensembles, memetic algorithms, etc. Obviously, mixing two algorithms with similar search biases will not provide any additional benefits. We need to mix search biases that are complementary to each other. Some initial work in mixing complementary algorithms and in maximizing the complementarity when selecting algorithms for mixing has led to very promising results [10]. In fact, the idea and benefit of mixing search bias were shown much earlier at the operator level, when different mutation operators were mixed together [8].

Co-evolution

When interacting with biologists, we—the evolutionary computation people—are often reminded what we call evolution is nothing like real biological evolution at all because there is no defined fitness function in biological evolution. What we call co-evolution is much closer to real biological evolution. Interestingly, there has been more research effort and applications in "evolution" than in "co-evolution" within the evolutionary computation community. Maybe it is time for us to study co-evolution in more depth.

One major characteristic of co-evolution lies in fitness evaluation. In co-evolution, the fitness of an individual depends on other individuals, not just on itself. This creates interesting but complex dynamics in evolution, which makes theoretical analysis of co-evolution very hard. In practice, co-evolution has often been used as a potential approach to achieving automatic divide-and-conquer of large and complex problems [12, 13], although more analyses of the experimental results are still needed.

Co-evolution has been used in both learning and optimization. It was studied extensively in the evolution of game-playing strategies and, to a lesser extent, in co-evolutionary optimization. However, all such work has been computational and empirical in nature. Very few theoretical results are available. Even the recently proposed theoretical framework for co-evolutionary learning [14] can only be seen as an initial step for thinking and evaluating co-evolution. The research in large-scale optimization using co-evolution [13, 15] also needs more support from theories. There are ample opportunities for future research in co-evolution.

Concluding Remarks

Evolutionary computation is still a very young research field. While it has overlaps with classical artificial intelligence, it has its own characteristics—population and stochasticity being two of them. The research in evolutionary computation theories has made a significant progress in the last decade, especially in computational time complexity analysis. However, such theories have mainly been limited to evolutionary combinatorial optimization. More fundamental research is needed.

There are many types of EAs, which share similar or the same essence. For example, hybrid EAs, memetic algorithms, and algorithms portfolios/ensembles all share the same essence of mixing different search biases in order to find a global optimum with little prior knowledge. The advantage of analyzing EAs in terms of search bias is that it allows us to select proactively complementary algorithms (or operators) in a hybrid algorithm. The initial work on algorithm portfolios consisting of complementary algorithms has demonstrated the potentials of this direction of research.

Co-evolution has not attracted as much attention as it should from the research community. Yet it is one of the most interesting topics in evolutionary computation, which is different from conventional EAs and also different from other algorithms in classical artificial intelligence and computer science. Recent work in global optimization using co-evolution and the new theoretical framework for co-evolutionary learning seem to have reignited people's interest in this topic. However, major progresses can only be made with significant advances in theoretical studies.

Evolutionary computation is a very broad field. This short article provides some thoughts on only a few limited topics. There are many other interesting challenges, such as optimization in a dynamic and uncertain environment, multi-objective optimization, online evolution, incremental evolution, etc.

References

[1] He, J. and Yao, X. Drift analysis and average time complexity of evolutionary algorithms. *Artificial Intelligence* 127, 1 (March 2001), 57–85.

[2] He, J. and Yao, X. From an individual to a population: An analysis of the first hitting time of population-based evolutionary algorithms. *IEEE Transactions on Evolutionary Computation* 6, 5 (October 2002), 495–511.

[3] Chen, T., Tang, K., Chen, G., and Yao, X. A large population size can be unhelpful in evolutionary algorithms. *Theoretical Computer Science*, p. DOI: 10.1016/j.tcs.2011.02.016, 2011.

[4] Neumann, F. and Witt, C. *Bioinspired Computation in Combinatorial Optimization: Algorithms and Their Computational Complexity*. Springer, Berlin, 2010.

[5] Lehre, P. K., and Yao, X. Crossover can be constructive when computing unique input-output sequences. *Soft Computing* 15, I9 (September 2011), 1675–1687.

[6] Lehre, P. K., and Yao, X. On the impact of mutation-selection balance on the runtime of evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, p. DOI: 10.1109/TEVC.2011.2112665, 2011.

[7] He, J., and Yao, X. Maximum cardinality matching by evolutionary algorithms. In *Proceedings of the 2002 UK Workshop on Computational Intelligence (UKCI'02)*, (Birmingham, UK, September 2002), 53–60.

[8] Yao, X., Liu, Y., and Lin, G. Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation* 3, 2 (July 1999), 82–102.

[9] Hart, W. E., Krasnogor, N., and Smith, J.E. (eds.) *Recent Advances in Memetic Algorithms*. Springer, Berlin, 2004.

[10] Peng, F., Tang, K., Chen, G., and Yao, X. Population-based algorithm portfolios for numerical optimization. *IEEE Transactions on Evolutionary Computation* 14, 5 (October 2010), 782–800.

[11] Tasgetiren, M. F., Suganthan, P. N., and Pan, Q. K. An ensemble of discrete differential evolution algorithms for solving the generalized traveling salesman problem. *Applied Mathematics and Computation* 215, 9 (January 2010), 3356–3368.

[12] Khare, V., Yao, X., and Sendhoff, B. Multi-network evolutionary systems and automatic problem decomposition. *International Journal of General Systems* 35, 3 (June 2006), 259–274.

[13] Yang, Z., Tang, K., and Yao, X. Large scale evolutionary optimization using cooperative coevolution. *Information Sciences* 178, 15 (August 2008), 2985–2999.

[14] Chong, S. Y., Tino, P., and Yao, X. Measuring generalization performance in co-evolutionary learning. *IEEE Transactions on Evolutionary Computation* 12, 4 (August 2008), 479–505.

[15] Li, X. and Yao, X. Cooperatively coevolving particle swarms for large scale optimization," *IEEE Transactions on Evolutionary Computation* 16, 2 (2012), 210-224.

About the Author

Xin Yao is a Professor of Computer Science at the University of Birmingham, UK. He is an IEEE Fellow, a Distinguished Lecturer of IEEE Computational Intelligence Society and a Distinguished Visiting Professor at the University of Science and Technology of China. He was a former editor-in-chief (2003-08) of *IEEE Transactions on Evolutionary Computation*. His research interests include evolutionary computation and neural network ensembles. His work won the 2001 IEEE Donald G. Fink Prize Paper Award, 2010 IEEE Transactions on Evolutionary Computation Outstanding Paper Award, 2010 BT Gordon Radley Award for Best Author of Innovation (finalist), and 2011 IEEE Transactions on Neural Networks Outstanding Paper Award.

DOI: 10.1145/2369296. 2369298