Evolutionary Computation: An Overview and Recent Trends

Günther Raidl*

Institute of Computer Graphics and Algorithms
Vienna University of Technology
Favoritenstraße 9–11/1861, 1040 Vienna, Austria
raidl@ads.tuwien.ac.at

1 Introduction

Evolutionary computation is a steadily increasing research discipline covering computer algorithms that are inspired by principles of natural evolution. The three main mechanisms that drive evolution forward are reproduction, mutation, and natural selection (i.e., the Darwinian principle of survival of the fittest). In the biological world these mechanisms enable life-forms to adapt to a particular environment over successive generations. The camel's hump, the eagle's eye, the dolphin's sonar, the human brain – they all can be seen as solutions to environmental problems that were generated by evolution. Evolutionary algorithms adopt these mechanisms of natural evolution in simplified ways and breed progressively better solutions to a wide variety of complex optimization and design problems.

A classical *genetic algorithm*, as it has been introduced by Holland [9], is shown in Fig. 1. It maintains a population of encoded candidate solutions to a given optimization problem. The initial solutions are usually created either at random or by simple construction heuristics. A stochastic selection process, which chooses better solutions with higher probabilities, is applied to obtain a set of parents. A new generation of candidate solutions is then derived by recombining these parents and mutating the offsprings. During recombination, new solutions are created out of the properties (genetic information) contained in the parents. Mutation typically performs just small random modifications.

Beside of genetic algorithms, evolution strategies [15], and evolutionary programming [8] emerged almost independently. Despite their superficial differences, these

*This work has been supported by the Austrian Science Fund (FWF) under grant P16263-N04.

```
procedure Genetic Algorithm
begin
  t ← 0;
  initialize(P(t));
  evaluate(P(t));
  while (not termination-condition) do
    t ← t + 1;
    Q_s(t) ← select(P(t − 1));
    Q_r(t) ← recombine(Q_s(t));
    P(t) ← mutate(Q_r(t));
    evaluate(P(t));
  done
end
```

Figure 1: Pseudo-code of a canonical genetic algorithm as introduced by Holland [9].

approaches follow the same spirit and share the same basic template. Differences lie mainly in the way how candidate solutions are represented and how selection, recombination, and mutation are actually implemented. While in original genetic algorithms solutions were often encoded as binary vectors, evolution strategies were mainly applied to continuous parameter optimization and represented solution candidates directly as vectors of real values. Evolutionary programming originally utilized finite state machines for representing solutions.

A fourth class of evolutionary algorithms, which is actually an offspring of genetic algorithms, became popular when Koza introduced *genetic programming* [10]. Here, candidate solutions are tree structures representing executable computer programs. Such a candidate program is evaluated by interpreting it and assigning a fitness value according to how well the program solves the problem at hand. First exemplary applications included symbolic regression and toy problems like following an artificial food path on a grid.

Nowadays the distinction between these approaches is blurring, particularly as far as representation and operators are concerned. Most researchers agree that candidate solutions should be represented in whatever way best suits the problem at hand. For example for real-valued parameter optimization problems, the parameter values are usually directly stored (in contrast to using a binary encoding), and a normal-distribution-based mutation, which was originally a particular feature of evolution strategies, is applied, even when the general framework conforms to a classical genetic algorithm. In addition, several further approaches have emerged that adopt other mechanisms from nature; for example, *ant colony optimization* [6] mimicks the behavior of a colony of ants finding the shortest path from the nest to a location containing food, *particle swarm optimization* is motivated by the social behavior of organisms such as bird flocking and fish schooling [4], and *artificial immune system methods* are based on principles of immune system defenses against viruses [5].

There are several advantages distinguishing evolutionary algorithms (EAs) from other optimization and problem solving techniques:

- Simple EAs do not require any in-depth mathematical understanding of the problems to which they are applied.
- Consequently, such EAs are relatively cheap and quick to implement.
- EAs are open to changes of the problem and can in general cope well with additional constraints, noisy, inaccurate, and incomplete data.
- Unlike many other methods, when EAs are implemented on parallel hardware they make efficient use of the available power in a very natural way.

On the other side, most researchers agree that simple evolutionary algorithms usually cannot compete with more sophisticated optimization algorithms that are tailored towards a specific problem and exploit certain features of it. This led to the development of many hybrid approaches, were an evolutionary algorithm is combined with problem-specific variation operators, "intelligent" encodings, and/or clever local improvement or repair algorithms. In fact, most of today's state-of-the-art EA-based approaches for important practical problems are relatively complex, problem-specific hybrid systems. The possibility of effectively combining an EA-framework with other techniques is therefore often perceived as another important strength.

For introductory books to evolutionary computation, see [7, 2]. A more extensive survey can be found in [1]. Newer concepts in evolutionary computation are overviewed in [4].

2 Recent Trends

During the last years research in evolutionary computation and the application of EAs to real-world problems, have steadily and significantly expanded. The following paragraphs summarize some most remarkable trends from the author's perspective.

Extended hybrid architectures. In particular with respect to solving discrete optimization problems, classical simple EAs are known to be typically less efficient than other state-of-the-art algorithms that are specifically tailored to the problems. As already mentioned above, EAs are therefore often combined with problem-specific algorithms. Beside this, also hybrids of EAs and other metaheuristics such as local search, simulated annealing, tabu search, guided local search, variable neighborhood search, and variable depth search are nowadays widely used, and for many hard combinatorial optimization problems, such approaches are today's leading algorithms for finding superior solutions to large problem instances in limited time. Such combinations are also called *memetic algorithms* [13]. Typically, the EA's primary responsibility is to take care of diversification and coverage of all major areas of the search space, while the embedded other metaheuristic is mainly responsible for intensification of the search, thus, the fine-tuning of the most promising candidate solutions.

Some highly promising work has also been done in combining EAs with exact optimization techniques including dynamic programming, branch-and-bound, and the large palette of (integer) linear programming techniques. In these hybrid systems, the exact technique may work as a subordinate of the EA in order to solve smaller subproblems, the EA may act as subordinate of the exact technique (e.g., to obtain better bounds), or both strategies can run in a parallel or interleaved way, exchanging information and stimulating each other. In the latter two strategies, the exact approach's property of being, in principle, able to prove optimality or to provide a quality guarantee can be retained. See [14] for an overview to this research direction.

Application domains. New applications where EAs prove to be well-suited tools are continously found. In particular bioinformatics, image analysis, signal processing, telecommunication, hardware optimization, and artificial music and art represent currently booming application domains. Dedicated workshops, such as the European EvoWorkshops or the different tracks at the leading conferences in evolutionary computation document this (see below).

Multiobjective optimization. In real-world optimization applications, it is often hard to formulate the optimization goal as a scalar function. Typically, there are several criteria or objectives, and not unusually, these objectives stay in conflict with each other. As an example, in the design of an automobile an engineer may wish to maximize crash resistance for safety and minimize weight for fuel economy. Simply combining the different associated objective functions in a linear way is usually unsatisfactory. Instead, one is interested in a so-called Pareto optimal set of solutions, i.e., any solution that cannot be improved with respect to one objective without worsening the situation with respect to the other objectives. Special strategies are therefore needed to deal with such multiobjective optimization problems. Since EAs work on populations of candidate solutions, they represent a promising basic framework for multiobjective optimization. In the last few years many variants and extensions of classical EAs have been developed and were demonstrated to be superior techniques for this purpose. For an overview on evolutionary multiobjective optimization, see [3].

Estimation of distribution algorithms (EDAs). In this relatively new class of EAs, a probabilistic model is used to represent promising solutions. A new generation of candidate solutions is derived by sampling the probabilistic model. These solutions are evaluated, and a selected subset is used to update and refine the probabilistic model. The simplest form of an EDA is the *univariate marginal distribution algorithm* (UMDA). In it, the variables representing a solution are considered independent and the probabilistic model is a vector of associated probabilities. More sophisticated EDAs also consider dependencies among the variables and use, for example, Bayesian networks as probabilistic models. For several problems, EDAs have shown to be able to outperform classical simple EAs. See [12] for an overview.

A particularly well-known and often applied variant of EDAs are *ant colony optimization* (ACO) algorithms [6]. They are often combined with problem-specific local search and have been successfully applied to a variety of combinatorial optimization problems such as vehicle routing or routing in networks.

Genetic Programming. Since its beginning in the early 90s, much progress has also been made with respect to genetic programming. This class of genetic algorithms that uses programs encoded as tree-structures as candidate solutions has shown to be able to automatically produce results that are competitive to patented or patentable human inventions [11]. Such successful applications are reported for the design of electronic circuits for analog filters and controllers, the development of quantum algorithms, and certain problems in bioinformatics.

Theory. Research in the theory of evolutionary computation has also made significant progress. Today, various theoretical models and results help to understand under which conditions an EA works well or is supposed to perform only poorly. The spectrum of algorithms which could be rigorously analyzed, including the derivation of tight bounds for the expected run-times and the expected quality of obtained solutions, has significantly increased. Nevertheless, there are still many open questions, and despite the achievements, theory is today unfortunately still far away from precisely predicting or fully describing the expected behavior of a more complex hybrid EA when applied to a hard problem from real world. See e.g. [16] for recent theoretical results.

3 Scientific Events

The largest scientific conference in the area of evolutionary computation is the *Genetic and Evolutionary Computation Conference* (GECCO), which is annually held in the United States and receives a continuously growing number of submissions (this year about 560). The second largest annual conference is the *IEEE Congress on Evolutionary Computation* (CEC). It has so far been held in the United States and Asia and will be organized in Edinburgh, U.K., this year. A smaller international event dedicated to theoretical developments in evolutionary computation is the *Foundations of Genetic Algorithms Conference* (FOGA).

In Europe, we have the bi-annual conference *Parallel Problem Solving from Nature* (PPSN) and the annual *EuroGP/EvoCOP/EvoWorkshops* event. The latter is a collection of two conferences dedicated to genetic programming (EuroGP) and metaheuristics for combinatorial optimization (EvoCOP) and a series of smaller workshops oriented towards specific application domains.

All the mentioned events peer-review submitted papers and publish accepted papers in their proceedings.

References

[1] T. Bäck, D. B. Fogel, and Z. Michalewicz. *Handbook of Evolutionary Computation*. Oxford University Press, New York, 1997.

[2] T. Bäck, D. B. Fogel, and Z. Michalewicz. *Evolutionary Computation 1 + 2*. Institute of Physics Publishing, 2000.

[3] C. A. C. Coello, D. A. V. Veldhuizen, and G. B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, 2002.

[4] D. Corne, M. Dorigo, and F. Glover. *New Ideas in Optimisation*. McGraw-Hill, Berkshire, England, 1999.

[5] L. N. de Castro and J. I. Timmis. *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer, 2002.

[6] M. Dorigo and T. Stützle. *Ant Colony Optimization*. MIT Press, 2004.

[7] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Springer, 2003.

[8] L. Fogel. *Artificial Intelligence Through Simulated Evolution*. John Wiley and Sons, 1966.

[9] J. H. Holland. *Adaption in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, 1975.

[10] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. The MIT Press, Cambridge, MA, 1992.

[11] J. R. Koza, M. A. Keane, M. J. Streeter, W. Mydlowec, J. Yu, and G. Lanza. *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*. Kluwer Academic Publishers, 2003.

[12] P. Larrañaga and J. A. Lozano. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Springer, 2001.

[13] P. Moscato. Memetic algorithms: A short introduction. In D. Corne et al., editors, *New Ideas in Optimization*, pages 219–234. McGraw Hill, 1999.

[14] J. Puchinger and G. R. Raidl. Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. In *Proceedings of the First International Work-Conference on the Interplay Between Natural and Artificial Computation*, LNCS. Springer, to appear 2005.

[15] H. P. Schwefel. *Numerical Optimization of Computer Models*. John Wiley, 1981.

[16] A. Wright, M. Vose, K. De Jong, and L. Schmitt, editors. *Foundations of Genetic Algorithms Conference 2005 (FOGA-8)*, LNCS. Springer, to appear 2005.