Algorithmica (2016) 75:425–427
DOI 10.1007/s00453-016-0157-4

Guest Editorial: Theory of Evolutionary Computation

Benjamin Doerr[1] · **Carsten Witt**[2]

Received: 14 April 2016 / Accepted: 26 April 2016 / Published online: 5 May 2016

Evolutionary algorithms are known for being easy-to-use optimization methods for all kinds of optimization problems. This has led to a large amount of empirical data on their working principles and on how to most suitably design these algorithms. However, in the last 20 years it became more and more evident that a true understanding and reliable rules for parameter choices can only be obtained in conjunction with theoretical means in the same flavor as in the classic algorithms field.

In this special issue, we collect four remarkable results on the theory of evolutionary computation. They were selected among the most significant contributions to the theory track of the 2014 *Genetic and Evolutionary Computation Conference (GECCO)*. The authors of these selected works were invited to significantly extend and polish their results and then present them in the exact and mathematical style of the journal *Algorithmica*. These papers underwent at least two rounds of careful reviewing according to the high standards of the journal before being accepted and now represent high-quality and trustworthy presentations of recent first-class research results.

The paper *Runtime Analysis of Non-Elitist Populations: From Classical Optimisation to Partial Information* by Dang and Lehre (http://link.springer.com/article/10.1007/s00453-015-0103-x) presents a powerful tool for the runtime analysis of population-based randomized search heuristics that use non-elitist selection mechanisms. General upper bounds on the runtime are obtained through a division of the search space into levels and estimations of transition probabilities. This tool significantly generalizes traditional techniques such as fitness-based partitions. The tool is

✉ Benjamin Doerr
doerr@lix.polytechnique.fr

[1] Laboratoire d'Informatique (LIX), École Polytechnique, 91128 Palaiseau, France

[2] DTU Compute, Technical University of Denmark, 2800 Kongens Lyngby, Denmark

then used in a rarely considered but important setting: only partial information on the fitness value is available, e.g., as some arguments are not evaluated. In this scenario, its shown that population-based evolutionary algorithms still can find an optimum given a sufficiently large population and small mutation probability.

The paper *Robustness of Populations in Stochastic Environments* by Gießen and Kötzing (http://link.springer.com/article/10.1007/s00453-015-0072-0) analyzes the role of populations in stochastic optimization. More precisely, it is assumed that the objective function is subject to noise, leading to stochastic errors in its evaluation. On classical benchmark functions, such noise makes optimization by the simple $(1+1)$ EA hillclimber infeasible even in exponential time. Interestingly, the use of parent and offspring populations of only logarithmic size turns the algorithm into an efficient one. The results are obtained by state-of-the art drift analysis techniques.

Under the term *drift analysis* a number of tools are subsumed that all allow to translate information on the progress an algorithm makes in one iteration into information about the time the algorithm takes to reach a particular goal. These tools are among the most powerful and most used in the runtime analysis of evolutionary algorithms. While multiplicative drift not only gives statements on the expected runtime, but also bounds that hold with high probability, such high-probability statements were much harder to obtain in the situation of additive drift. In his paper *Concentration of First Hitting Times under Additive Drift*, Timo Kötzing (http://link.springer.com/article/10.1007/s00453-015-0048-0) provides a number of new results that give high-probability bounds in the presence of additive drift.

One of the most important problems in evolutionary computation is to understand which problems can be solved efficiently by evolutionary algorithms. On the theory side, a number of positive results exist, both for artificial problems and classic optimization problems. In his paper *Superpolynomial Lower Bounds for the (1 + 1) EA on Some Easy Combinatorial Problems*, Andrew Sutton (http://link.springer.com/article/10.1007/s00453-015-0027-5) points out three classic problems that are all polynomial-time solvable, but for which simple evolutionary algorithms with high probability need more than polynomial time. He also shows how to modify the standard evolutionary approach so that the problems are solved in polynomial time.

The initialization phase of evolutionary algorithms seems to be the part most neglected by research so far. Theoretical work often tries to argue that a random initial search point behaves not too differently from an initial search point with average fitness, that is, that the runtimes differ only by lower order terms. This often true observation is made very precise for two classic test problems in the paper *The Impact of Random Initialization on the Runtime of Randomized Search Heuristics* (http://link.springer.com/article/10.1007/s00453-015-0019-5) by Doerr and Doerr. They show that for the OneMax and LeadingOnes test functions, the expected runtimes of the randomized local search heuristic and the $(1+1)$ EA differ only by a constant number of iterations when comparing the initialization with a random search point and with one of average fitness.

Dynamic optimization, that is, optimization in the presence of dynamically changing problem data, is one of the younger but very important streams in evolutionary computation. The general hope is that evolutionary algorithms, in particular through the use of populations that may store more information about the search space than just the current optimum, are well-suited to cope with changing inputs. In their paper *MMAS versus Population-Based EA on a Family of Dynamic Fitness Functions* (http://link.springer.com/article/10.1007/s00453-015-9975-z), Lissovoi and Witt show this effect on the maze example function. More precisely, they show that a $(\mu + 1)$ EA can easily optimize the maze function over an alphabet of size μ, whereas a population size of $\mu - 1$ would not suffice. However, they also show that the MMAS ant colony optimizer can efficiently optimize all these fitness functions, suggesting that ant colony algorithms might be even better suited for dynamic optimization.

We thank all authors for their submissions, our reviewers for their careful reading and detailed comments, and last but not least the *Algorithmica* team and editor-in-chief Ming-Yang Kao for their great support.

Benjamin Doerr and Carsten Witt, March 2016