Algorithmica (2018) 80:1575–1578
https://doi.org/10.1007/s00453-017-0379-0

Preface to the Special Issue on Theory of Genetic and Evolutionary Computation

Timo Kötzing[1] · **Dirk Sudholt**[2]

Published online: 3 October 2017

Evolutionary algorithms (EAs) are randomized search heuristics that can be employed to solve complex optimization problems, including multimodal or highly constrained problems. EAs work by mimicking principles from natural evolution: maintaining a collection of possible solutions (a population) and iteratively creating variants of the individuals (the offspring) and then choosing a new set of individuals for the next iteration (selection). EAs are popular because they represent general-purpose optimizers that can be easily applied to various problems, even in cases where little or no in-depth knowledge about the problem is available. In order to guide practitioners devising new and effective algorithms, theoretical computer scientists employ methods from the field of randomized algorithms to analyze the working principles of EAs with mathematical rigor. Key questions concern the impact of parameter choices (such as, for example, the offspring size or the choice of variation operators) as well as foundational work on developing powerful analysis methods. The theory track of the annual ACM *Genetic and Evolutionary Computation Conference (GECCO)* is the first tier event for advances in this direction.

In this special issue six selected papers from the 2016 edition of the GECCO theory track are collected, each one of them carefully revised and extended to meet the high quality standards of *Algorithmica*.

Black box complexity is a complexity notion where computation steps are not considered costly, but instead evaluations of a hidden (and to-be-optimized) fitness

✉ Timo Kötzing
Timo.Koetzing@hpi.de

Dirk Sudholt
d.sudholt@sheffield.ac.uk

1 Chair of Algorithm Engineering, Hasso Plattner Institute, Potsdam, Germany

2 Department of Computer Science, University of Sheffield, Sheffield, UK

function are counted. Thus, the black box complexity gives a measure for how hard it is to optimize a given problem class in terms of queries to the fitness function. Most notably, this concept provides very general *lower* bounds, applying to a wide range of algorithms. The paper *The (1 + 1) Elitist Black-Box Complexity of LeadingOnes* considers the class of algorithms which keep only the best-so-far solution and construct one offspring from that. The considered problem class is the permutation- and bit-invariant LeadingOnes function class. The authors show a tight lower bound for the black box complexity of $\Omega(n^2)$ by employing a new information-theoretic technique and a careful analysis of how information can be stored in the bit string during the optimization process. This new technique is the main contribution of this paper: not many tools for proving lower bounds are known, adding one to the toolbox is a major achievement. Thus, it is not surprising that the paper won the best paper award of the GECCO 2016 Theory track.

Many classic algorithms for optimization focuses on the case where the fitness function is *unimodal*, that is, following improving moves necessarily leads to the global optimum. Modern approaches to optimization try to deal with the problem of *multimodal* optimization where local optima have to be overcome by somehow crossing the fitness valley. The paper *How to Escape Local Optima in Black Box Optimization: When Non-Elitism Outperforms Elitism* considers two properties of the fitness valley, its depth and its length, and analyzes which of these properties has an impact on how hard or easy it is for certain algorithms to cross that valley. Interestingly, while the crossing time of the classic $(1 + 1)$-EA depends only on the *length* of the valley, for the Metropolis algorithm and the SSWM algorithm (both of which can step down into the valley) the crossing time depends mainly on the valley's *depth*. This notable result is one of few analyses of how non-elitist search heuristics can overcome local optima.

The Maze function is a simple benchmark for dynamic optimization in the Boolean hypercube. In this benchmark, the optimum moves around the hypercube, and each such transition consists of a long phase of oscillation where the old and the new optimal point alternate in having the best fitness. This allows (for some) algorithms to track the optimum. Latest in a sequence of papers on the Maze function and on how different algorithms can or cannot follow the optimum, *The Impact of a Sparse Migration Topology on the Runtime of Island Models in Dynamic Optimization* extends previous work on how island models can track local optima. The authors show that frequent migration in a topology with small diameter leads to losing the optimum, while a sufficiently large diameter (for example in a ring topology of at least $c \log n$ islands, where $c > 0$ is a sufficiently large constant) allows for successfully tracking the optimum. In particular, this gives an example where sparse migration topologies are advantageous. These results are complemented by analyses of settings where migration occurs less frequently (where denser migration topologies are better) and by experimental results on parameter settings currently not accessible to a theoretical investigation.

The $(1 + (\lambda, \lambda))$ Genetic Algorithm is one of very few genetic algorithms for which it could be proven that crossover leads to an asymptotic speedup. It is the first, and so far only unbiased genetic algorithm proven to solve all OneMax-like functions in time $o(n \log n)$. In *Optimal Static and Self-Adjusting Parameter Choices for the* $(1 + (\lambda, \lambda))$ *Genetic Algorithm* the authors study the impact of the population size λ,

the mutation probability p, and the crossover bias c. Contributions are as follows: first, the authors improve previous upper bounds for fixed parameters $p = \lambda/n, c = 1/\lambda$, and arbitrary choices of $\lambda \leq n$. This refined bound matches a previously known lower bound, allowing the authors to derive optimal values for the population size λ in the setting of $p = \lambda/n, c = 1/\lambda$. Next, the authors question whether other choices of p and c could lead to a better performance. They prove a lower bound for the whole three-dimensional parameter space spanned by λ, p, and c, confirming that it is optimal to fix $p = \lambda/n, c = 1/\lambda$. Along the way, the authors present new methods and insights for analyzing multi-dimensional parameter spaces where parameters interact in unforeseen ways. Finally, the authors replace a static choice of λ with a self-adjusting scheme using a discrete analogue of the one-fifth success rule. They prove that the self-adjusting $(1 + (\lambda, \lambda))$ Genetic Algorithm needs only linear time on OneMax, which is faster than any static parameter setting.

Many evolutionary algorithms use an offspring population: they create many search points in one generation. The simplest such algorithm is the $(1 + \lambda)$ Evolutionary Algorithm, shortly $(1 + \lambda)$ EA. In *Optimal Mutation Rates for the* $(1 + \lambda)$ *EA on* OneMax *Through Asymptotically Tight Drift Analysis* the authors refine existing results on the run time of the $(1 + \lambda)$ EA for arbitrary mutation rates c/n, for constant $\lambda, c > 0$. They analyze the progress made in one generation, called *drift*, and derive exact expressions for the drift for each possible fitness value of the current search point. They further present a refined variable drift theorem for lower bounds, allowing them to derive very tight upper and lower bounds on the expected optimization time. This shows a surprisingly close relationship between drift and expected optimization times: once the drift is known, the expected optimization time can be estimated very closely, up to small-order terms. This finding, as well as their refined drift theorem, are of independent interest and may prove useful in further studies. The authors conclude that for small problem sizes n, the optimal mutation rate is slightly larger than the asymptotically optimal one of $1/n$, however absolute differences are small.

Almost all run time analyses of evolutionary algorithms consider the binary space $\{0, 1\}^n$. The paper *Static and Self-Adjusting Mutation Strengths for Multi-valued Decision Variables* considers variants of the OneMax function defined over a search space $\{0, 1, \ldots, r-1\}^n$ of multi-valued decision variables, where each of n variables takes values within $\{0, 1, \ldots, r-1\}$. The main research question is: how to design efficient mutation operators for the multi-valued domain? If every mutated variable is set to a uniform random value within $\{0, \ldots, r-1\}$, the expected run time is $\Theta(nr \log n)$. If each mutated variable is only changed by ± 1, the expected run time reduces to $\Theta(nr + n \log n)$. The authors also consider a harmonic distribution where a step of length $\pm i$ is chosen with a probability inversely proportional to the step length i, and prove an upper bound of $\Theta(n \log(r)(\log n + \log r))$. A final, natural mechanism is to self-adjust the mutation strength based on the success of previous iterations. This leads to an expected run time of $\Theta(n(\log n + \log r))$, which is best possible among all dynamic mutation strengths.

We hope that with this special issue we further increase the interest of the general algorithms research community into evolutionary computation methods. We thank all authors for their submissions, our reviewers for their helpful and detailed comments,

and last but not least the *Algorithmica* team and the editor-in-chief Ming-Yang Kao for their great support.

Timo Kötzing and Dirk Sudholt
Potsdam and Sheffield, August 2017