Benchmarking discrete optimization heuristics: from building a sound experimental environment to algorithm configuration

Ye, F.

Chapter 1

Introduction

A fundamental research field of optimization is discrete optimization, which is prevalent in real-world applications, ranging from operation research (e.g., scheduling [33] and logistics [9, 66]) to other fields of computer science (e.g., neural architecture search [64]). It is also of vital importance in theoretical studies, e.g., NP-hardness of optimization problems [6, 66, 70] and the analysis of algorithmic complexity of optimization algorithms [37, 40, 49, 84, 90, 91, 110].

Many discrete optimization problems are hard-to-solve, and various algorithms are proposed in the literature, e.g., [9, 10, 33, 70, 72, 99, 102, 152], to solve different problems. In this thesis, we focus only on the study of *evolutionary computation (EC)* methods, which have been widely and successfully applied for discrete optimization [9, 33, 70, 72, 99, 152]. An increasing number of algorithms such as genetic algorithms (GAs) [161], estimation of distribution algorithms (EDAs) [103], genetic programming (GP) [101], etc., are being proposed to solve different complex problems. Meanwhile, we lack a clear conclusion about the performance of these algorithms across different types of problems. Therefore, one key challenge of our research community is *to develop guidelines on which algorithms to favor for which kinds of problems.* Another important challenge is to *enhance our understanding of performance limits of EC methods,* so as to gain insight into the potential of further improving their design. Different approaches have been conducted to address these two key challenges, ranging from theoretical analysis of the algorithms [51, 86, 124] to a practical comparison of the algorithms on real-world optimization problems [134, 164].

1.1 Benchmarking in Optimization

The theory-oriented approach and the approach of real-world optimization are conducted to understand algorithms' performance. Meanwhile, benchmarking studies can offer an intermediate approach associating these two approaches. The two approaches complement each other, as they can help us gain insight into algorithms' behaviour via different approaches, and there is much untapped potential in bounding these approaches together. For example, an algorithm proposed using one approach can help solve problems in another approach by generating hypotheses tested on the approach where the algorithm was originally proposed, which can be easily achieved between the benchmarking study and real-world optimization by applying algorithms assessed on benchmarks to real-world optimization problems. Benchmarking may also benefit theoreticians by enhancing mathematically-derived ideas into techniques being broadly applicable in practical optimization. It also constitutes a catalyst for formulating new research questions for theoretical research and the real-world optimization domain.

The recent discussion [12] in the EC community has summarized that benchmarking studies can aim for: (1) *Assessment of Problems and Algorithms.* Benchmarking can provide empirical analyses of algorithms' performance across different optimization problems, addressing the questions of how the increasing number of algorithms compare across different optimization problems. Consequently, we can select the "winner" algorithms for competitions. On the other hand, it can also help us assess the optimization problems and illustrate the algorithms' behaviour during the optimization process; (2) *Sensitivity of Performance.* Benchmarking can assess the sensitivity of algorithms with response to different problem instances. It will also benefit parameter tuning by helping us learn the impact of the parameters for an algorithm on certain problem instances. Moreover, it supports characterizing algorithms' performance by problem instance features and vice versa; (3) *Performance Exploration.* Benchmarking can produce data for machine learning tasks to explore promising algorithms for a given problem. Benchmarking data can also be used for automatic algorithm design, selection, and configuration; (4) *Complementing and Inspiring Theoretical Study.* The experimental results of benchmarking studies can help valid theoretical results. Moreover, the benchmarking results may inspire theoretical works; (5) *Reproducibility and Algorithm Development.* Benchmarking projects can support us in verifying the reproducibility of a given program. Also, understanding algorithms' behavior addresses the question of how the underlying design of algorithms can be used to solve different types of optimization problems. Furthermore, benchmarking may inspire novel designs

of algorithms.

We will demonstrate the potential of benchmarking by showing in this thesis how it inspires new theoretical results and the design of new algorithms. Also, we will present how we can gain insights into the behaviour of the automatic algorithm configuration tools based on experiments on classic theory-oriented problems.

1.2 Scope of the Thesis

We will apply the benchmarking approach in this thesis to investigate the performance of EC methods on different problems. With respect to the algorithms to be investigated, we focus on *iterative optimization heuristics* (IOHs). The IOHs aim to find the optimal solution for a problem minimizing or maximizing $f : S \to \mathbb{R}, x \mapsto f(x)$ by searching for new solution candidates iteratively. For each iteration, one or a set of solution candidates $\{s_1, s_2, \ldots, s_\lambda\} \in S$ are generated. The algorithm terminates when a specific criterion is met, e.g., the time budget is reached, or a solution with the required quality is found. The class of IOHs subsumes evolutionary algorithms (EAs), (Quasi-)Monte Carlo algorithms, swarm intelligence, differential evolution, EDAs, efficient global optimization, Bayesian optimization, local search variants (for example, first/steepest ascent and variable neighbourhood search), etc. As for the optimization problems, we focus on *pseudo-Boolean optimization problems*, a subset of discrete optimization.

1.3 The Demand for a New Benchmarking Environment

In the context of discrete optimization, several attempts to construct widely accepted benchmarking software have been undertaken [81, 132], but these are typically restricted to certain problem classes (often classical NP-hard problems such as Satisfiability (SAT) [81], Traveling Salesperson Problem (TSP) [132], etc.) without attempting to generate a set of scalable or generalizable optimization problems. In addition, many frameworks strongly focus on constructive heuristics, which are assumed to have access to the problem instance data (in contrast to black-box optimization heuristics, which implicitly learn about the problem instance only through the evaluation of potential solutions). The few attempts to create a sound benchmarking platform for discrete black-box optimization heuristics, e.g., Weise's optimization benchmarking

platform [158], did not receive significant attention from the scientific community.

In December 2018, Facebook announced its benchmarking environment for black-box optimization [131]. Though their Nevergrad platform comprises a few discrete problems such as TSP, shifted sphere function [145], etc., it mainly focuses on noisy continuous optimization. Another famous benchmarking project, the COmparing Continuous Optimizers (COCO) [79] software constitutes well-established and widely recognized software for benchmarking derivative-free black-box optimization heuristics. The COCO software is under constant development. Apart from its well-known BBOB benchmark set [80], it also offers noisy, multi-objective [149], and mixed-integer [148] problem collections. While COCO has been designed to analyze IOHs on different types of problems, its designers have chosen to pre-select these problems that users can test their algorithms on. Benchmarking new problems with COCO requires substantial knowledge of its software design, and is therefore quite time-consuming. Also, it requires additional work using COCO to extend the performance statistics and visualizations.

To break down barriers between different tools and unify research ideas from different domains, we wish for a sound benchmarking environment that allows algorithms to be tested on a wide range of problems and visualize algorithm behavior with statistical analysis. Moreover, implementing problems or algorithms of existing benchmark projects shall not require much effort to be integrated into this new benchmarking software. The design of such benchmarking environment shall take in account the components of optimization algorithms to be compared, the types of optimization problems, the performance measures used to evaluate algorithms, and other possible techniques to present algorithms' behaviour. Considering these components, we have developed the **benchmarking software IOHprofiler** with a modular structure, which provides the environment to perform the study of this thesis.

1.4 Research Questions

This thesis concentrates on an overarching research question:

How can benchmarking studies benefit theoretical analysis on the optimization problems and the practical study of algorithm design?

From considering *how to build benchmarking software* to presenting *what we gain from benchmarking studies*, we provide comprehensive study cases to illustrate valid answers

for this question.

As building the IOHprofiler benchmarking software, we discuss the questions:

1. *Which components shall an applicable benchmarking pipeline take into account?* To answer this question, we have addressed the sub-questions of what problems and algorithms we are interested in, how to collect the raw data of experiments, what performance measures we can apply for analyzing algorithms' performance, and how to visualize these results. In addition, technique issues of software development such as the choices of programming language, the efficiency of implementation, and version control are also involved in developing our IOHprofiler benchmarking software.

Benefiting from IOHprofiler, we can compare different algorithms on a set of benchmarking problems for the following open question:

2. *How do various evolutionary algorithms perform on different problems, and what is the impact of the parameters and the operators?* In this thesis, we investigate how the population size affects the performance of $(1+\lambda)$ EAs on OneMax and LeadingOnes, how the optimal mutation rates adjust during the optimization process for OneMax and LeadingOnes, if/how the uniform crossover is helpful on LeadingOnes, and what the promising configurations of a $(\mu+\lambda)$ GA are for different types of problems.

The benchmarking results motivate future research on the topic of parameter tuning, answering the following questions:

3. *How to interpolate local and global search during optimization?* This question is motivated by observing how randomized local search (RLS) and different EAs perform at different stages of the optimization process for OneMax and LeadingOnes

4. *What is the impact of the cost metric (i.e., the expected running time (ERT) and the area under the empirical cumulative distribution function curve (AUC)) for the performance of algorithm configuration methods?* We investigate the performance of the promising configurations obtained using algorithm configuration methods. Moreover, the results help us understand the behaviour of algorithm configuration methods.

5. *What are the next steps for algorithm configuration?* We leverage our benchmarking data of various GAs for the topic of dynamic algorithm selection. Our

investigation on the performance of different dynamic algorithm policies reveals the potential topics for the future work on dynamic algorithm selection.

1.5 Our Contributions

Overall, this thesis involves three topics: benchmarking discrete optimization algorithms, empirical analyses of evolutionary computation, and automatic algorithm configuration. The objective is benchmarking EAs on discrete optimization for the selection and the design of better optimizers.

In practice, we start with **building the IOHprofiler benchmark software.** IOHprofiler consists of two main parts: IOHexperimenter and IOHanalyzer. IOHexperimenter provides a platform for programming experiments, and IOHanalyzer performs post-processing of algorithm performance data. IOHprofiler is the cornerstone of the study of this thesis. It supports us in testing algorithms on a wide range of problems and allows us to perform and visualize the statistical analysis on algorithms' performance. Benefiting from the functionalities of IOHprofiler, we can systematically work on the benchmark study in the following chapters.

Our initial benchmarking work focuses on the two classic problems OneMax and LeadingOnes. We **study the impact of mutation rate and population size** on the $(1 + \lambda)$ EA. We find that the $(1 + \lambda)$ EA benefits from small λ for OneMax. However, the value of λ does not significantly affect the ERT values for LeadingOnes. In addition, **we observe that crossover can help for LeadingOnes** by testing a $(\mu + \lambda)$ genetic algorithm with different crossover probabilities. We find that as μ increases, the value of the optimal crossover probability increases, while for fixed μ, its value decreases with increasing problem dimension.

The work of analyzing EAs and local search algorithms on OneMax and LeadingOnes clearly shows that local search is preferable at some (in our use-case, late) stages of the optimization process, whereas larger *mutation strengths* of the EAs are beneficial at other stages. Adaptive choice allows us to leverage complementarity. Therefore, we analyze in this thesis a smooth way of interpolating between local and non-local search by **proposing a new *normalized bit mutation***. Experimental results show improvement in using the normalized bit mutation compared to using either local search or EAs.

Our benchmarking study is also performed on more benchmarking problems provided by IOHprofiler. Twelve heuristics and various configurations of the $(\mu+\lambda)$ GA are tested. **We investigate how crossover and mutation interplay with each**

other and the impact of population size. The obtained result provides us, for different benchmark problems, the promising settings of mutation rate, crossover probability, and population size. The benchmark data also inspires us towards the work of automatic algorithm configuration and dynamic algorithm selection.

We apply three AC techniques: iterated racing (Irace) [111], mixed-integer parallel efficient global optimization (MIP-EGO) [155], and mixed-integer evolutionary strategies (MIES) [109], to configure the $(\mu + \lambda)$ GA for two different objectives, i.e., ERT and AUC, respectively. The AC methods aim at automatically finding the best configuration (i.e., the optimal parameter setting and operator choice) of an algorithm for the given problem. However, first, we need to decide on a *cost metric* as the objective of the configuration task. We investigate the impact of minimizing ERT and maximizing AUC as the cost metric, respectively. **Our results suggest that even when interested in expected running time performance (i.e., ERT), it might be preferable to use *anytime performance* measures (i.e., AUC) for the configuration task. We also observe that tuning for ERT is much more sensitive with respect to the budget that is allocated to the target algorithms.**

Apart from applying algorithm configuration for static settings of evolutionary algorithms, we are also interested in the dynamic settings. **We leverage our benchmark data of static algorithms for the study of *dynamic algorithm selection*.** The study inspires us to focus on the automatic detection of the timing of switching algorithms and efficient *warm-start* strategies of the switching.

1.6 Structure of the Thesis

Chapter 2 provides relevant background for this thesis. In particular, we provide here an introduction to optimization, EAs, the performance measures, and the benchmark problems that are used in this thesis. Chapter 3 then introduces the IOHprofiler benchmarking software. Chapter 4 presents our first use-case of benchmarking on the two classic problems, OneMax and LeadingOnes. The proposed standard normalized bit mutation is also introduced in Chapter 4. Chapter 5 continues to benchmark twelve heuristics and various configurations of the $(\mu + \lambda)$ GA on more benchmark problems provided by IOHprofiler. The topic of AC is discussed in Chapter 6, where we apply the three techniques, Irace, MIP-EGO, and MIES, to configure the $(\mu + \lambda)$ GA. In Chapter 7, we leverage the benchmark data of Chapter 5 for the study of dynamic algorithm selection. The thesis is summarized in Chapter 8, which briefly

recalls the main contributions and discusses future research topics inspired by our work.

1.7 Software and Publications

This thesis is based on the following works.

1.7.1 Software and Documentation

A key contribution is the IOHprofiler environment that is available on the following websites.

- IOHprofiler wiki page: https://iohprofiler.github.io.
- IOHprofiler Github https://github.com/IOHprofiler.
- IOHanalyzer web-based interface: https://iohanalyzer.liacs.nl.

1.7.2 Journal Publications

1. Carola Doerr, Furong Ye, Naama Horesh, Hao Wang, Ofer M Shir, and Thomas Bäck. Benchmarking discrete optimization heuristics with IOHprofiler. *Applied Soft Computing*, 106027. Elsevier, 2020.

   This paper contributes to Section 5.1. It describes our benchmarking results of twelve heuristics on the twenty three problems provided by IOHprofiler.

2. Furong Ye, Carola Doerr, Hao Wang, and Thomas Bäck. Automated Configuration of Genetic Algorithms by Tuning for Anytime Performance. *IEEE Transactions on Evolutionary Computation*. IEEE, 2022.

   This paper contributes to Chapter 6. It presents our work of tuning the $(\mu + \lambda)$ GA for minimizing ERT or maximizing AUC on the twenty five problems provided by IOHprofiler, using the three AC techniques, Irace, MIP-EGO, and MIES.

3. Hao Wang, Diederick Vermetten, Furong Ye, Carola Doerr, Thomas Bäck. IOHanalyzer: Detailed Performance Analyses for Iterative Optimization Heuristic. *ACM Transactions on Evolutionary Learning and Optimization*, in press. ACM, 2022.

This paper contributes to Section 3.3. It introduces the IOHanalyzer module of IOHprofiler in detail.

4. Jacob de Nobel*, Furong Ye*, Diederick Vermetten, Hao Wang, Carola Doerr, and Thomas Thomas Bäck. IOHexperimenter: Benchmarking Platform for Iterative Optimization Heuristics. *arXiv preprint arXiv:2111.04077*. 2021. (Under revision of *Evolutionary Computation Journal*)(*These authors contributed equally to this work.)

   This paper contributes to Section 3.2, which introduces the IOHexperimenter module of IOHprofiler in details.

1.7.3 Peer-reviewed Conference Publications

1. Carola Doerr, Furong Ye, Sander van Rijn, Hao Wang, Thomas Bäck. Towards a theory-guided benchmarking suite for discrete black-box optimization heuristics: profiling $(1 + \lambda)$ EA variants on OneMax and LeadingOnes. *In Proc. of Genetic and Evolutionary Computation Conference (GECCO'18)*, 951–958. ACM, 2018.

   This paper contributes to Section 4.1. It publishes our benchmarking results of the $(1 + \lambda)$ EAs on OneMax and LeadingOnes. The results motivated a refined analysis for the optimization time of the $(1 + \lambda)$ EA on LeadingOnes.

2. Furong Ye, Carola Doerr, and Thomas Bäck. Interpolating Local and Global Search by Controlling the Variance of Standard Bit Mutation. *In Proc. of IEEE Congress on Evolutionary Computation (CEC'19)*, 2292–2299. IEEE, 2019.

   This paper contributes to Section 4.2. It investigates how the mutation rate affects the performance of the $(1+\lambda)$ EAs on OneMax and LeadingOnes. The standard normalized bit mutation is proposed in this work.

3. Furong Ye, Hao Wang, Carola Doerr, and Thomas Bäck. Benchmarking a $(\mu+\lambda)$ Genetic Algorithm with Configurable Crossover Probability. *In Proc. of Parallel Problem Solving from Nature (PPSN'20)*, 699-713. Springer, 2020.

   This paper contributes to Sections 4.3 and 5.2. It investigates the impact of the crossover probability for the $(\mu+\lambda)$ GA. This work shows that crossover

can be helpful for LeadingOnes and inspires us with the study of dynamic crossover probability.

4. Furong Ye, Carola Doerr, and Thomas Bäck. Leveraging Benchmarking Data for Informed One-Shot Dynamic Algorithm Selection. *In Proc. of Genetic and Evolutionary Computation Conference (GECCO'21), Companion Material*, 245–246. ACM, 2021.

   This paper contributes to Chapter 7. It publishes our result of leveraging benchmark data in Section 5.2 for dynamic algorithm selection.

1.7.4 Other Documentation

1. Carola Doerr, Hao Wang, Furong Ye, Sander van Rijn, Thomas Bäck. IOHprofiler: A benchmarking and profiling tool for iterative optimization heuristics. *arXiv preprint arXiv:1810.05281*. 2018.

   This article contributes to Chapter 3. It presents the general overview of the IOHprofiler software.