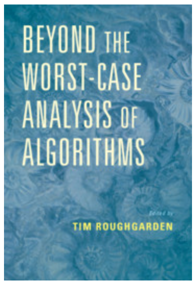
Review of [1]



Beyond the Worst-Case Analysis of Algorithms
edited by **Tim Roughgarden**

Cambridge University Press, 2021
USD 68.99, Hardcover, 704 pages

Reviewed by **Jonathan Katz**
Dept. of Computer Science, University of Maryland

1 Overview

When we traditionally evaluate algorithms, we do so based on their performance on *all possible inputs*. That is, we expect an algorithm for some problem to solve that problem for every possible input, and we measure the efficiency of the algorithm by looking at its performance on all inputs of a given size. This *worst-case* approach is the perspective taught to students when they are first introduced to programming, throughout undergraduate algorithms courses, and in most of complexity theory. And it is the viewpoint typically taken even when considering more "advanced" algorithms like randomized algorithms (which should solve the problem with high probability for any input), approximation algorithms (which should approximate the solution for any input), or online algorithms (which must be competitive with the best offline algorithm for all possible inputs).

Yet there are several examples of algorithms that have very good performance in practice even though their worst-case guarantees are quite poor. (Some examples are described below.) Is it possible to develop theoretical frameworks "beyond worst-case analysis" that can justify the use of such algorithms or help develop improved algorithms with superior real-world performance?

The present volume, edited by Tim Roughgarden, contains 29 surveys of different approaches for addressing exactly this question. Each survey is written by an expert (or experts) in a particular domain, and taken together they provide an approachable and thorough introduction to the diverse techniques for going beyond worst-case analyses of algorithms. Most of the surveys also contain extensive references for additional reading.

The Introduction, written by Tim Roughgarden, provides a robust defense (in case one is needed!) of the value of going beyond worst-case analyses. In particular, Roughgarden provides four examples of instances where worst-case analysis fails to adequately explain real-world performance:

1. In the context of linear programming, the simplex method does well in practice but may run for exponential time on certain inputs. But the inputs on which it runs for exponential time are "pathological"—indeed, even finding a sequence of inputs that forces the algorithm to run for exponential time was an open research question in the 1970s. In contrast, the ellipsoid method has a polynomial-time bound on its worst-case running time but performs

[1]©2023 Jonathan Katz

much worse than the simplex method in practice. What theoretical approach[2] can be used to justify using the simplex method instead of the ellipsoid method?

2. A worst-case analysis of online algorithms (which must operate in an "online" fashion as inputs arrive, with incomplete knowledge of future inputs) for page caching shows that on certain input sequences the "least recently used" (LRU) policy can result in a 100% cache-miss rate. Yet the LRU policy is the one favored by systems architects, due to the empirical observation that it performs well on most real-world access sequences. How can the practical effectiveness of the LRU-based algorithm be accounted for?

3. Clustering (under most objective functions that people care about in practice) is NP-hard. Yet in practice there are several algorithms that output good clusterings when run on real-world data. How can this discrepancy be explained?

4. The real-world success of neural networks for solving supervised learning problems is hard to ignore. But it is difficult to provide theoretical explanations of both the observed convergence time of gradient-descent methods, as well as the effectiveness of overparameterized neural networks in correctly labeling unobserved data. What guidance can theorists offer here?

I hope the reader will enjoy coming up with their own solutions the above conundrums before reading some possible answers below!

2 Summary of the Book

The book itself, consisting of the remaining 29 chapters, is divided into six parts, four which can be said to cover "techniques" and two containing surveys focusing on particular "applications." A non-exhaustive list of the chapters in the different parts follows.

Part One. The first part of the book can roughly be said to deal with algorithmic analyses that still follow a worst-case approach, but with generalized ways of measuring efficiency (beyond merely expressing it as a function of the input size). This includes a chapter on *parameterized algorithms*, where the idea is to measure the running time of algorithms based on some parameter k of the input (and not only the input size), a chapter on *instance-optimal algorithms* that (up to a constant multiplicative factor) perform better than any other algorithm on every input, and a chapter on *resource augmentation* where, roughly, the performance of an algorithm using some amount of resources is compared to an optimal algorithm using fewer resources.

Part Two. The next part of the book can be viewed as focusing on meaningful ways to constrain the set of inputs for which an algorithm is analyzed (i.e., either the algorithm does not need to be correct for inputs outside some set of interest, or the running time of the algorithm for those inputs is not considered); this allows for excluding "pathological" inputs on which an algorithm might do poorly. This part has three chapters. The first two constrain the input space by restricting attention to *stable* inputs, under different notions of stability—one based on properties of the input

[2]Of course, one might argue that no theoretical justification is needed if we have empirical evidence that the simplex method is better. This is certainly unsatisfying to a theorist who wants to "explain" the observed behavior. More generally, taking that stance provides no assurance to a user that the inputs on which they run the simplex algorithm will not be exactly those pathological inputs that lead to a long running time, and it gives no guidance to researchers for developing improvements to the simplex method.

instances themselves, and the other based on properties of the solutions to those instances. The third chapter restricts attention to algorithms that are tailored for *sparse instances* in the context of streaming algorithms, compressed sensing, and linear algebra.

Part Three. The chapters in the third part of the book consider what might be called "average-case analysis," where there is some distribution on inputs and algorithms are measured by their expected performance rather than their worst-case performance. This part includes a chapter on *distributional analysis*, where a simple distribution over inputs is assumed, as well as a chapter on *planted models*, where an input is chosen from some distribution but that input can then be modified in a (restricted) adversarial fashion. A counterpart to planted models considered in another chapter is the *random-order model*, where an adversary first specifies an input and then that input is randomly shuffled before being given to an algorithm. (That model makes the most sense in the context of online algorithms.) This part concludes with a chapter on *self-improving algorithms* that do not know the input distribution in advance, but are instead supposed to "learn" how to perform well for the (unknown) input distribution as they are presented with more and more input samples drawn from that distribution.

Part Four. The next part of the book focuses on *smoothed analysis.* Roughly, smoothed analysis considers a worst-case selection of the input, but assumes the input is randomly perturbed before it is given to an algorithm. This is a compelling framework in settings where input instances may not be exact, e.g., they may be based on real numbers obtained from physical measurements that anyway have some inherent imprecision or uncertainty. While the three chapters here could also fit into the previous part, the field of smoothed analysis has become important enough to justify giving these chapters their own dedicated section of the book.

Part Five. The final two parts of the book look at various applications that can benefit from analytical approaches to algorithms that go beyond a worst-case analysis. Part Five considers applications in the areas of machine learning and statistics. The first two chapters look at solving machine-learning and statistical problems in the presence of (adversarial) *noise.* Later chapters examine conditions under which stochastic gradient descent and overparameterized models have provable guarantees. Other chapters focus on *nearest-neighbor search and classification*, computing *low-rank tensor decompositions*, and *instance-optimal algorithms for distribution testing/learning.* While several of these chapters could have fit in previous parts based on the techniques being used, it makes sense to put them together in Part Five since they all related to learning.

Part Six. The final section consists of a "grab bag" of applications of some of the techniques from earlier parts. There is a chapter on alternatives to worst-case competitive analysis of online algorithms, an important chapter on SAT-solving algorithms (which work well in practice even as they solve instances of an NP-complete problem), a chapter on auctions from the perspective of algorithmic game theory, and one on algorithms for restricted classes of graphs suitable for modeling social networks.

2.1 Some Solutions to the Motivating Problems

Based on the above outline, the reader may already be able to guess some possible solutions to the motivating problems discussed at the beginning of this review. The book proposes the following (of course, other solutions may exist!):

1. Arguably one of the greatest successes of smoothed analysis was its application to analysis of the running time of the simplex method. (Spielman and Teng received the Gödel prize in 2008 for that result.) Roughly, it can be shown that the simplex method has polynomial smoothed complexity with respect to Gaussian perturbations of the input. This can explain why the simplex algorithm runs quickly on inputs encountered in the real world.

2. The LRU page-caching algorithm can be profitably analyzed using a parameterized approach where performance is measured as a function of the *locality* of the input sequence. (Locality measures how many distinct pages can be requested in a given number of consecutive requests.) Already in Chapter 1, it is shown that the LRU algorithm is essentially optimal when measured that way.

3. Although clustering is NP-hard, the hardest instances are essentially those that don't cluster well in the first place; for such instances, clustering is not an appropriate goal, anyway. As Roughgarden puts it in Chapter 1, "clustering is hard only when it doesn't matter." Conversely, then, clustering algorithms tend to perform well when they are applied to problems for which clustering makes sense as a solution concept.

4. Approaches for explaining the real-world performance of supervised learning algorithms are explored in Part Five. Roughly, gradient descent can be shown to perform well for inputs satisfying a natural criterion that typically holds in practice. Understanding the success of neural networks is currently an active area of research, but some plausible explanations are discussed in the chapter on overparameterized models.

3 Evaluation

I enjoyed reading this book, and enthusiastically recommend it for graduate students or researchers looking for an introduction to this active area of research. Overall, the quality of the chapters is high, the topics covered are compelling, and good pointers to the literature are given. While the book does suffer from some of the drawbacks of most edited volumes (e.g., inconsistent notation, different styles, a relative lack of integration between different chapters), it is clear that some effort was made to alleviate these issues to the extent possible.

I don't work in the field of algorithms, but I do teach undergraduate algorithms from time to time. In reading the book, it struck me that a good chunk of the material here could be taught at the undergraduate level, and might spark interest in students (and even instructors) bored by the well-trodden algorithms and applications that constitute a typical course in algorithms. While the current book could be used by a instructor (with significant effort) to develop undergraduate lecture materials or as part of a graduate course, it would be a challenge to do so because the book is simply not written with that goal in mind. (And I would not recommend assigning it to your average undergraduate student, though perhaps a sharp, theoretically inclined student could make some headway.) Perhaps the next generation of algorithms textbooks will incorporate some of the material covered in this book in a more-accessible manner suitable for undergraduates.