The Book Review Column[1]

by William Gasarch
Department of Computer Science
University of Maryland at College Park
College Park, MD, 20742
email: gasarch@cs.umd.edu

Welcome to the Book Reviews Column. We hope to bring you at least three reviews of books every month. In this column three books are reviewed.

1. **Approximation Algorithms for NP-hard Problems** , Edited by Dorit S. Hochbaum. Reviewed by Randeep Bhatia and Yoram J. Sussmann. This is a collection of articles (by different authors) on approximating a large variety of NP-hard problems.

2. **Selected Papers on Computer Science** by Donald E. Knuth. Reviewed by Samir Khuller. This is a collection of articles on computer science by Donald E. Knuth. They are intended for a wide audience.

3. **Communication Complexity** by Eyal Kushilevitz and Noam Nisan. Reviewed by William Gasarch. This is a monograph on communication complexity.

Review of
Approximation Algorithms for NP-hard Problems[2]
Editor: Dorit S. Hochbaum
Publisher: PWS, 1997

Reviewed by: Randeep Bhatia and Yoram J. Sussmann
University Of Maryland, College Park.

1 Overview

This book is the first comprehensive guide to the field of approximation algorithms. It is organized as a series of more or less stand-alone chapters on topics ranging from inapproximability proofs to the development of approximation algorithms for several important classes of NP-hard problems. Most of the chapters are devoted to a survey of a specific class of problems; however there are also several chapters that focus on a specific issue or technique in the field, such as hardness of approximation or randomization.

2 Summary of Contents

Approximation Algorithms for Scheduling. *By Leslie A. Hall.* The algorithms presented in this chapter span three decades of research beginning with Graham's List Scheduling, widely believed to be the first heuristic analyzed for its worst case performance, to the recent work on designing approximation algorithms for scheduling problems with min-sum objective functions. In addition the chapter is a comprehensive survey of many basic scheduling problems such as those in which the objective is to minimize the maximum completion time, lateness etc. under various

job constraints and machine environments. The author also surveys approximation algorithms for various job-shop scheduling problems. The survey nicely illustrates the use of techniques like rounding based on LP relaxations, randomization etc. for the design of approximation algorithms for many hard scheduling problems.

Approximation Algorithms for Bin Packing: A Survey. *By E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson.* The bin-packing problem has a long history in the field of approximation, appearing early on in the study of worst-case performance guarantees and in proving lower bounds on the performance of online algorithms. It has also figured prominently in the development of average-case analysis of algorithms. This survey covers the development of algorithms for the one-dimensional bin-packing problem, and includes discussions of both worst-case and average-case performance. Off-line and online algorithms are both covered as well.

Approximating Covering and Packing Problems: Set Cover, Vertex Cover, Independent Set, and Related Problems. *By Dorit S. Hochbaum.* Linear programming, first used for the analysis of the greedy heuristic for set cover, has proved to be an important tool for the development and analysis of approximation algorithms. The algorithms in this survey rely heavily on the use of LP relaxation and duality for their performance. To illustrate this technique, a variety of covering-type problems are studied. The method is introduced with a discussion of several approaches to the set cover problem. The presentation is then extended to include related problems such as vertex cover, independent set, and graph coloring. The survey also includes a discussion of integer programs with two variables per inequality. This class of problems helps shed light on the benefit of using LP methods to attack covering and packing problems.

The Primal-Dual Method for Approximation Algorithms and its Application to Network Design Problems. *By Michel X. Goemans and David P. Williamson.* The primal-dual method has proven to be an effective tool in the design of efficient algorithms for combinatorial optimization problems. Recently the authors in conjunction with some other researchers were able to extend the power of the primal-dual method to the design of good approximation algorithms for a wide variety of NP-Hard problems. In this chapter the authors reconstruct the steps that lead to these developments and also illustrate the power of this technique for the design of near optimal approximation algorithms for a host of network design problems such as the min multicut problems, survivable network design problems etc.

Cut Problems and their Application to Divide and Conquer. *By David B. Shmoys.* This chapter is a survey on the design of approximation algorithms for cut problems. It is shown that efficient algorithms for the cut problems can be used to design efficient approximation algorithms for a number of combinatorial optimization problems based on the paradigm of divide and conquer. The author surveys the best known approximation algorithms for the multicommodity flow problems and the multicut problems. It is shown that good approximation algorithms for a balanced cut problem can be utilized as a "divide" subroutine in the design of efficient divide and conquer approximation algorithms for a number of problems such as the minimum cut linear arrangement problem, minimum chordal extension problem and the storage time product problems. Some other highlights of the results presented are the approximate max-flow min-cut theorems for multicommodity flow and approximation algorithms for the min-cost feedback arc set problem in directed graphs.

Approximation Algorithms for Finding Highly Connected Subgraphs. *By Samir Khuller.* The problem of finding graphs of a desired connectivity is fundamental in designing networks to survive failures. For example, the problem of finding a subgraph that connects all vertices (a 1-connected subgraph) is simply the minimum spanning tree problem. Such a subgraph is easily susceptible to a single edge or vertex failure, however. In general one may desire to find a spanning subgraph that can withstand 1 or more failures. This chapter covers both vertex- (where the number of vertices that can fail is given) and edge- (where a specified number of edges may fail) connectivity problems for both weighted and unweighted graphs. The problem of augmenting the connectivity of a graph is also studied as well as the problem of finding a minimum strongly connected spanning subgraph of a strongly connected directed graph.

Algorithms for Finding Low Degree Structures. *By Balaji Raghavachari.* Problems that involve finding subgraphs of low degree satisfying given properties have many practical applications. In telecommunications applications, switches have a limited capacity, and degree constraints have to be considered when designing the network. In broadcast networks where we want to minimize work done at each site the maximum degree becomes important as well. The simplest such problem is the minimum-degree spanning tree problem (MDST), where we are asked to find a spanning tree of a graph G whose maximum degree is minimized over all spanning trees of G. This results covered in this chapter include algorithms for the MDST problem and other related problems including finding minimum-degree Steiner trees, minimum degree 2-connected spanning subgraphs, and minimum-weight bounded degree spanning trees in edge-weighted graphs. The performance of a simple local search heuristic for approximating minimum degree problems is also analyzed for several problems.

Approximation Algorithms for Geometric Problems. *By Marshall Bern and David Eppstein.* Many graph problems that are hard to solve in general have better solutions in a geometric context; an example is the recent PTAS for the Euclidean traveling salesman problem. This chapter surveys approximability and hardness results for a number of problems in the plane. Five general areas are covered: the traveling salesman problem (TSP), the Steiner tree problem, minimum-weight triangulation, clustering, and separation. Unfortunately the PTAS for Euclidean TSP came too late to be included in this chapter. Problems in this chapter include the Euclidean Steiner trees problem, min-weight triangulations, several clustering problems, and the k-MST problem.

Various Notions of Approximations: Good, Better, Best, and More. *By Dorit S. Hochbaum.* Approximation algorithms are often grouped into classes according to how close an approximation to the optimum they provide. This chapter discusses a number of classifications, including FPAS, PTAS, constant approximations, and additive error approximations. Numerous examples are included to illustrate the different levels of approximability, including some covering and packing problems and clustering problems.

Hardness of Approximations. *By Sanjeev Arora and Carsten Lund.* A survey on approximation algorithms would not be complete without the recent results on the inapproximability of many NP-Hard problems. These results rely on a probabilistic characterization of the NP class by the so-called PCP theorem. The authors provide an excellent survey of almost all the major inapproximability results without requiring of the reader any knowledge of the complicated algebraic techniques used to establish the PCP theorem. It is shown that NP-Hard problems can be grouped into four classes based on the approximation ratio that is hard to achieve for them. Six canonical problems are used for this purpose, the most basic being MAX-3SAT. It is shown that

MAX-3SAT is $1+\epsilon$ hard to approximate for some $\epsilon > 0$. Reductions from MAX-3SAT are then used to establish all the other inapproximability results.

Randomization Approximation Algorithms in Combinatorial Optimization. *By Rajeev Motwani, Joseph (Seffi) Naor and Prabhakar Raghavan.* The design of approximation algorithms for many combinatorial optimization problems share a three step approach: formulate the problem as an integer program, relax certain constraints of the integer program to make it efficiently solvable and then round the solution of the relaxed program to a feasible solution of the original program. This chapter is a comprehensive survey of approximation algorithms which make use of randomization for achieving the rounding in the third step. The technique is illustrated for two different kinds of relaxations (used in the second step) of integer programs: linear programming relaxations and semidefinite programming relaxations. Application of this approach is presented for the design of efficient approximation algorithms for problems such as Graph Coloring, finding Max-Cuts, Covering and Packing, MAX-SAT etc.

The Markov Chain Monte Carlo Method: An Approach to Approximate Counting and Integration *By Mark Jerrum and Alistair Sinclair.* The Markov Chain Monte Carlo Method (MCMCM) is presented as a general paradigm for the design of efficient approximation algorithms for a wide variety of fundamental computational problems. These problems tend to be complete for the **#P** class and hence computationally even harder than NP-complete combinatorial optimization problems. Given Ω a very large but finite set of combinatorial structures and π a probability distribution on Ω the MCMCM provides an algorithm for sampling an element of Ω at random according to the probability distribution π. The techniques described in this chapter have proved useful in designing efficient approximation algorithms for problems such as counting matchings of all sizes in a graph, counting the permanent of a given matrix and estimating the volume of a convex body in high dimensional spaces. It is also shown that widely used heuristics such as *simulated annealing* are just special cases of the MCMCM.

Online Computation. *By Sandy Irani and Anna R. Karlin.* An online algorithm works without any knowledge of the future. Such an algorithm is said to be competitive if its performance is close to that of an optimal algorithm that sees all its input in advance. This chapter is an excellent survey of the techniques used in the competitive analysis of online algorithms. These techniques are illustrated using three basic problems: the paging problem, the k-server problem and metrical task systems. The author surveys various deterministic and randomized online algorithms for these problems. The results described include the $(2k-1)$-competitiveness of the work-function algorithm for the k-server problem and the exponential function technique for virtual circuit routing.

3 Opinion

This text provides a good reference for both experienced and beginning researchers working in the area. It covers most major results and topics in the field and provides a good summary of many major techniques. It could be used as a text for a graduate class; however, the book is not sequential and the instructor would have to fill in the gaps and provide a general introduction. The chapters are self-contained, but in some cases to thoroughly understand the results one has to resort to the original papers, since important details are left out. However, other chapters are a delight to read.

Book Review for "Selected Papers on Computer Science" by Donald E. Knuth

Samir Khuller
Department of Computer Science
University of Maryland, College Park, MD 20742
samir@cs.umd.edu

"Selected papers on computer science" by Donald E. Knuth [3] is a collection of lectures and published papers aimed at a wide audience. A detailed narrative, replete with illustrations and examples, this collection focuses on three principle themes: the centrality of algorithms to computer science, the importance of combining theory and practice, as well as the historical development of the field.

Addressing the first of these issues, Knuth argues that computer science is essentially about algorithms and algorithmic thought. As he puts it, "Algorithms are the life-blood of computer science ... the common denominator that underlies and unifies the different branches". Defining algorithms as concepts that exist independent of programs, even though programming languages are used to express algorithms, Knuth states that "a person well trained in computer science knows how to deal with algorithms, how to construct them, manipulate them, understand them, analyze them". In fact, in Knuth's view, an attempt to comprehend mathematical concepts as algorithms results in a far deeper understanding than might ordinarily result. However, even as Knuth underscores the importance of algorithms as a "general purpose mental tool" he simultaneously identifies them as a point of difference between computer science and mathematics. Through a series of examples, he demonstrates that while mathematicians are content once they prove that a certain object that satisfies certain properties *exists*, a Computer Scientist tries to learn how to *compute* this object. Such a viewpoint, has led to fundamental advances in Computer Science[3]. Simpler proofs, as well as alternative proofs and a better understanding of mathematical structures can be achieved when one puts on a "Computer Science hat".

The second theme that Knuth focuses on is the importance of combining theory and practice, something that in his opinion can have a significant payoff for the field as a whole. As he puts is "the world gets better theoretical tools as it gets better practical tools". In this context, Knuth specifically points to his experiences in developing METAFONT, and argues that this very practically oriented project generated many interesting new problems and thus had a very profound impact on his research. Some of these problems are discussed in Chapters 6–9. Some sample problems are: solving the K-centers problem on a line (a minor modification of the Frederickson and Johnson [2]) algorithm solves this problem), digitizing straight lines, hyphenating words etc.

Pervasive throughout the book, are historical snippets and comments about the historical development of computer science and the impact of the Persian scholar Abu 'Abd Allāh Muhammad ibn Mūsā al-Khwārizmī, whose textbook on arithmetic had a significant influence for about 500 years. Chapters 11-12 take the reader through a grand tour of ancient Babylonian number systems, Algorithms, Von Neumann's first computer program. A chapter that I really enjoyed reading was on Knuth's experiences with programming the IBM 650 and its nuances.

[3] In my opinion, Edmonds classic paper on matching [1] which developed a polynomial time algorithm for maximum matching, is another such example, where the search for an efficient algorithm to find a perfect matching led to the definition of polynomial time computability. Hall's theorem gives necessary and sufficient conditions for the existence of a perfect matching, but does not immediately yield an algorithm to find one.

Knuth also devotes a chapter to the contributions of George Forsythe (Foresight?) whose leadership and dedication was to a large extent responsible for establishing the Computer Science Department at Stanford, as well as in establishing Computer Science as a discipline.

Include is a list of the chapters in the book.

List of Chapters

0) **Algorithms, Programs and Computer Science**

1) **Computer Science and its Relation to Mathematics**

2) **Mathematics and Computer Science: Coping with Finiteness**

3) **Algorithms**

4) **Algorithms in Modern Mathematics and Computer Science**

5) **Algorithmic Themes**

6) **Theory and Practice I**

7) **Theory and Practice II**

8) **Theory and Practice III**

9) **Theory and Practice IV**

10) **Are Toy Problems Useful?**

11) **Ancient Babylonian Algorithms**

12) **Von Neumann's First Computer Program**

13) **The IBM 650: An Appreciation from the Field**

14) **George Forsythe and the Development of Computer Science**

15) **Artistic Programming**

Opinion

I must admit that I hesitated before taking on the task of writing a book review, since (a) I had never done this kind of thing before, (b) I thought that reading an entire book to comment on would take a huge amount of time. After reading the first half of the book, on a pleasant sunday afternoon, I was very glad that I took on this task. This book is written for a very broad audience, and the writing style makes reading it a very stimulating experience. The book is accessible to a reader with some knowledge of computer science or mathematics (an undergraduate degree in computer science certainly suffices), but is excellent reading for graduate students and faculty as well.

The book [3] in the usual Knuth style, pays a lot of attention to detail, and is fascinating reading for students of Computer Science, interested in the historical development of the field; especially those who think the field is simply about the *Web* and *Java*! In all, I found the discussion immensely satisfying and further reinforced my belief that algorithms are extremely useful and play a role in many unusual and unexpected situations.

References

[1] J. Edmonds, "Paths, trees and flowers", *Canad. J. of Math.*, 17 (1965), pp. 449–467.

[2] G. N. Frederickson and D. B. Johnson, "Finding kth paths and p-centers by generating and searching good data structures", *Journal of Algorithms*, 4 (1983), pp. 61-80.

[3] D. E. Knuth, *Selected papers on Computer Science*, Cambridge Univ. Press, (1996).

Review of: **Communication Complexity** [4]
by Authors: Eyal Kushilevitz and Noam Nisan
Publisher: Cambridge University Press

Reviewed by W. Gasarch, University of Maryland at College Park

1 Overview

Assume that Alice has $x \in \{0,1\}^n$ and Bob has $y \in \{0,1\}^n$. They both want to know if $x = y$ but they want to exchange as few bits as possible. They can solve this problem with $n+1$ bits of communication (Alice sends Bob x, and Bob sends back 1 if $x = y$ and 0 otherwise.) The following questions come to mind

1. Can they do better than $n+1$ bits?
2. Can they do better if both of them can flip coins privately and they tolerate a small probability of error?
3. Can they do better if some referee flips coins publicly and they tolerate a small probability of error?

Communication complexity makes these questions (and many others) rigorous, and answers some of them. Many of the results are used to prove lower bounds on a variety of models such as VLSI, circuits, and decision trees. This is not an accident— much of communication complexity was developed with these goals in mind, and directly applied to them.

2 Summary of Contents

2.1 Chapters 1,2,3,4: Two Party Communication Complexity

Assume that Alice and Bob have unlimited computational power. Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$. Assume Alice has $x \in \{0,1\}^n$ and Bob has $y \in \{0,1\}^n$. Assume that, in all cases below, the goal is to have a protocol that ends with both Alice and Bob knowing $f(x,y)$. Our interest is in how many bits of information Alice and Bob need to exchange, hence refered to as 'the number of bits' Protocols are modeled by decision trees.

1. $D(f)$ is the number of bits required in a deterministic protocol to compute f.

[4] © W. Gasarch, 1997.

2. $N(f)$ is the number of bits required for a nondeterministic protocol for f (every time a nondeterministic choice is made that will be charged as one bit of communication). If $f(x,y) = 1$ then some reachable leaf will have Alice and Bob both thinking that $f(x,y) = 1$. If $f(x,y) = 0$ then all reachable leaves will have Alice and Bob both thinking $f(x,y) = 0$. (The book uses a different but equivalent definition.)

3. $R_\epsilon(f)$ is the number of bits needed in a protocol that allows both Alice and Bob to flip coins, but also allows a probability of error of $\leq \epsilon$. If the ϵ is unspecified then it is $\frac{1}{3}$. (Any constant less than $\frac{1}{2}$ would suffice. The choice of constant only changes the multiplicative constant.)

4. R_ϵ^{pub} is the number of bits needed in a protocol that allows a referee to flip coins that both Alice and Bob can look at. If the ϵ is unspecified then it is $\frac{1}{3}$.

5. Several other variants of randomized protocols are also defined.

In Chapters 1,2, and 3 (1) many techniques are developed to prove upper and lower bounds on $D(f)$, $N(f)$, $R_\epsilon(f)$, and R_ϵ^{pub} for a variety of f, and (2) many theorems comparing these classes to each other are given. Chapter 4 consists of advanced topics that draw on the knowledge of Chapters 1,2, and 3. We give a sampling of results from Chapters 1–4.

Upper and Lower Bounds

We define three functions below that will be referred to during this review. All of them are functions from $\{0,1\}^n \times \{0,1\}^n \to \{0,1\}$.

$$EQ(x,y) = \begin{cases} 1 & \text{if } x = y; \\ 0 & \text{if } x \neq y. \end{cases}$$

$$NE(x,y) = 1 - EQ(x,y).$$

1. $D(EQ) = D(NE) = n + 1$.

2. $N(EQ) = n + 1$. $N(NE) = 1 + \lceil \log n \rceil$.

3. $R(EQ) = O(\log n)$.

4. $R^{\text{pub}}(EQ) = O(1)$.

Theorems about Classes

For all the theorems below f is a function from $\{0,1\}^n \times \{0,1\}^n$ to $\{0,1\}$.

1. Let M be the $2^n \times 2^n$ matrix with rows and columns indexed by $\{0,1\}^n$. such that entry (x,y) is $f(x,y)$. Then $D(f) \geq rank(M)$. The proof involved linear algebra.

2. Let $g(x,y) = 1 - f(x,y)$. Then $D(f) = O(N(f)N(g))$. The proof is combinatorial.

3. $N(f) = \Omega(\log D(f))$. The proof is combinatorial. This is tight since $N(NE) = O(\log n)$ and $D(NE) = n$.

4. $R(f) = \Omega(\log D(f))$. The proof involves Alice and Bob actually exchanging bits that encode probabilities. This is tight since $R(EQ) = O(\log n)$ but $D(EQ) = n + 1$.

5. $R_{\epsilon+\delta}(f) \leq R_\epsilon(f) + O(\log n + \log \frac{1}{\delta})$. This is proven by a probabilistic argument.

Advanced Topics

1. Assume $p \in N$. Let $f^p(x_1, y_1, \ldots, x_p, y_p) = (f(x_1, y_1), \ldots, f(x_p, y_p))$. Clearly $D(f^p) \leq pD(f)$. Can we ever obtain $D(f^p) << pD(f)$. The same can be asked for $N(f)$ and $R(f)$. Very few general theorems are known; however, it is known that $R(EQ^{\log n}) = O(\log n \log\log n) << (\log n)R(EQ) = O(\log^2 n)$.

2. One can define an analog to P in communication complexity by defining a 0-1 valued function f to be in P^{cc} if it can be computed with poly-log bits of communication. One can define NP and R similarly. Results from Chapters 1,2,3 show that $P^{cc} = NP^{cc} \cap coNP^{cc}$, $P^{cc} \neq NP^{cc}$, and $P^{cc} \neq coNP^{cc}$. In Chapter 4 they show that the set $DISJ(x, y)$ is $coNP^{cc}$-complete. ($DISJ(x, y) = 1$ iff x and y, as sets represented by bit vectors, are disjoint.)

2.2 Chapters 5,6,7: Other Models of Communication

The models of communication used in Chapters 1–4 are simple. Because of the simplicity of the model, the results in those chapters are of limited use in obtaining lower bounds for other (non-communication based) models of computation. In Chapters 5,6, and 7 the book investigates models that are not quite as simple as those in Chapters 1,2,3, but can be used to obtain lower bounds in other fields of theory. We give some highlights of these models and the lower bounds they are used to prove. Since we will do this, we will not summarize Chapters 8–14 on applications.

Let $S \subseteq \{0,1\}^n \times \{0,1\}^n \times Z$. ($Z$ is any set.) Assume Alice has $x \in \{0,1\}^n$ and Bob has $y \in \{0,1\}^n$. Alice and Bob would like to find *some* z such that $(x, y, z) \in S$. If no such z exists then we do not care what Alice and Bob come up with. $D(S)$ is the number of bits Alice and Bob have to exchange to come up with some z. The complexity of relations can be tied very closely to the depth of a circuit. Let f be a monotone function. Let S be the relation

$$\{(x, y, i) \mid f(x) = 0 \wedge f(y) = 1 \wedge x_i = 0 \wedge y_i = 1\}.$$

Then $D(S)$ is exactly the same as the optimal depth of a monotone circuit for f. In Chapters 10 and 11 this relation, along with lower bounds for $D(S)$ for various S, are used to prove lower bounds for monotone circuit depth. The problems considered include s-t-connectivity, Matching, and Set Cover. The work on the Complexity of Relations was done for the purpose of obtaining lower bounds on circuits; hence its application to such is not surprising.

What happens if there are more than two people involved in a protocol for a function? We will deal with three people, but k people would be similar. Let $f : \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$. One could define the complexity of f in terms of one person having x, one having y, and one having z and seeing how many bits are needed to compute $f(x, y, z)$. That is is not what is done. Instead they let Alice have y, z, Bob have x, z, and Carl have x, y.

Each player takes turns writing on a blackboard that the other two can see. With this model one can easily define the multiparty complexity of f. There are some applications of the lower bounds on multiplayer complexity to circuits (in Chapter 11). This work was not done with applications in mind, hence there are fewer of them.

In all the models looked at so far the string that Alice got and the string that Bob got were predetermined. What happens if we look at different ways to partition the input? One can look at best cases, worst cases, and cases where some of the input is available to both parties. The results obtained here are applied to VLSI (Chapter 8), decision tree complexity (Chapter 9), data structures (Chapter 9), branching programs (Chapter 12), and other topics. The work on variable partitions was done for the purpose of obtaining lower bounds on VLSI, hence its applications there are not surprising. Its applications elsewhere might be surprising.

3 Opinion

This is a great book! The topics are well chosen and the presentation is excellent. Each concept is illustrated with one or two well chosen examples. In fact, the examples are an integral part of the book.

The book mostly uses combinatorics, linear algebra, and the probabilistic method. Hence it can be read by a bright undergraduate. The authors provide a table (page xii) to answer the question 'If I want to read about topic XXX then which chapters do I need to read?' Hence if your main interest is (say) Decision Trees, then you need only read Chapters 7 and 9.

The authors have chosen to omit some topics that would require harder mathematics (e.g., the connection to Mobius functions). They often give one or two examples, but no more. This approach has kept the book short and readable. There are few wasted words.

I spend most of my Thanksgiving and Christmas vacation (and some time in between) reading this book. That I wanted to speaks well of its choice of topics. That I was able to speaks well of its presentation.

4 Errata

There is a website of errata at http://www.cs.technion.ac.il/~eyalk/book.html