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

In this column we review the following books.

1. **Boolean Functions and Computation Models** by Clote and Kranakis. Review by R. Gregory Taylor. This monograph covers an wide range of topics including Boolean functions and Boolean circuit families, propositional proof systems, sequential and parallel models of computation, and function algebras.

2. **Selected Papers in Discrete Mathematics** by D. Knuth. Review by Carlos A.S. Oliveira. This book is a compilation of work published (plus some unpublished papers) by the author.

3. **Linear Optimization and Extensions – Problems and Solutions** by Dimitris Alevas and Manfred Padberg. Review by Carlos A.S. Oliveira. This book is a companion to another book on this topic; however, it stands alone as a good collection of problems and solutions in the field of linear optimization and linear programming.

4. **Introduction to the Design and Analysis of Algorithms** By Ananay Levitin. Reviewed by William Fahle. This is an algorithms book for an undergraduate course that is organized by technique rather than problem.

Books I want Reviewed

If you want a FREE copy of one of these books in exchange for a review, then email me at gasarchcs.umd.edu

Reviews need to be in LaTeX, LaTeX2e, or Plaintext.

Books on Algorithms

1. *A Course on Computational Number Theory* by Henri Cohen.

2. *Algorithms: Design Techniques and Analysis* by Alsuwaiyel.

3. *Computational Techniques of the Simplex Method* by Maros.

4. *Immunocomputing: Principles and Appliations* by Tarakanov, Skormin, Sokolova.

5. *Dynamic Reconfiguration: Architectures and Algorithms* by Vaidyanathan and Trahan.

Complexity Theory

1. *Computational Complexity: A Quantitative Perspective* by Marius Zimand

2. *Algebraic Complexity Theory* by Burgisser, Clausen, and Shokrollahi.

3. Classical and Quantum Computing (with C++ and Java Simulations) by Hardy and Steeb.

Books on Cryptography

1. *Elliptic Curves: Number Theory and Cryptography* by Larry Washington.
2. *Block Error-Correcting Codes: A Computational Primer* by Xambo-Descamps.

Combinatorics

1. *Tolerance Graphs* by Golumbic and Trenk
2. *Combinatorial Designs: Constructions and Analysis* by Stinson.

Modelling and Semantics

1. *Logic in Computer Science: Modelling and Reasoning about Systems* by Huth and Ryan.
2. *Semantic Integration of Heterogenous Software Specifications* by Martin Groβe-Rhode.

Misc

1. *Symbolic Asymptotics* by John Shackell.
2. *The Random Projection Method* by Vempala. (This book applies the method to algorithms and complexity and learning.)
3. *Alfred Tarski: Life and Logic* by Feferman and Feferman.

Review of[2]

Boolean Functions and Computation Models
by Peter Clote and Evangelos Kranakis
Published by Springer-Verlag, 2002, 601 pages

Reviewed by R. Gregory Taylor (Manhattan College)

This wide-ranging monograph covers a number of research areas including Boolean functions and Boolean circuit families, propositional proof systems, sequential and parallel models of computation, and function algebras. If anything links them all together, it is surely complexity.

Of the several available models for computing Boolean functions, Boolean circuits present themselves as a particularly interesting choice according to the authors. First, Boolean circuits have application within the field of parallel algorithm design. Second, lower bounds for Boolean circuits are closely related to the P =? NP question. Chapter 1 of the book under review covers the basics of Boolean functions and Boolean circuits while Chapter 2 takes up lower bounds.

Chapter 1 (Boolean Functions and Circuits). Classes of Boolean functions corresponding to various well-known complexity classes are defined. Thus NC^k is the class of functions $f : \{0,1\}^* \mapsto \{0,1\}$ whereby each member f_n of an associated family $\{f_n | f_n = f \upharpoonright \{0,1\}^n\}$ is computed by an $O(\log^k n)$-depth, $O(n^{O(1)})$-size Boolean circuit C_n with fan-in 2. The class AC^k is defined similarly substituting unbounded fan-in AND- and OR-gates for arbitrary gates with fan-in at most

[2]

2. (So AC^0 is the class of Boolean functions computed by constant-depth polynomial-size circuits with NOT-gates and with unbounded fan-in AND- and OR-gates.)

In Section 1.6 the topic is Boolean circuits and formal languages. It is shown that (the characteristic function of) any regular (context-free) language is in nonuniform NC^1 (AC^1). (Chapter 1 considers nonuniform circuit families only.)

Section 1.7 reviews circuits for arithmetic operations. Addition of just two n-bit integers is shown to be in AC^0, and their multiplication in NC^1. As for integer division, an algorithm based on Newton iteration and due to Brent puts it in NC^2 whereas a more complex technique involving Chinese remaindering and due to Beame, Cook, and Hoover improves this to NC^1. (The very recent result of Allender, Barrington, and Hesse putting integer division in uniform TC^0 is not covered.)

Section 1.8 takes up the design of circuits for arbitrary Boolean functions. First, there is the standard technique based upon disjunctive normal form. The existence of "universal circuits" is demonstrated, specifically, a size-$O(2^{2^n})$ circuit with multiple outputs such that any n-ary Boolean circuit is computed by one of them. It is then shown that, with such a universal circuit in hand, any n-ary Boolean function f is computable by a circuit with fan-in 2 and size $O(2^n/n)$ (Shannon's Method). Another technique due to Lupanov improves this to $2^n/n + o(2^n/n)$. If f happens to be symmetric, then a size $O(n)$ circuit is possible—still fan-in 2—and it is mentioned in passing that depth $O(\log n)$ may be assumed.

Section 1.9 shows that, given Boolean circuit C of fan-in 2 but arbitrary fan-out having n input- and m output-gates, it is possible to construct an equivalent fan-in-2 fan-out-2 circuit C' whose size and depth are bounded above by linear functions in $size(C)$, $depth(C)$, n, and m.

Section 1.10 focuses on Boolean formulæ (fan-out-1 circuits). If fan-in 2 is also assumed, then such a formula is a binary tree with an associated *leafsize* (number of input-gates or number of occurrences of sentence letters). It is shown (Theorem 1.10.2(1) due to Spira) that the minimal depth of a fan-in-2 Boolean circuit computing $f : \{0,1\}^* \mapsto \{0,1\}$ lies between $\log_2(L(f)+1)$ and $k \cdot \log_2(L(f)+1)$, where $L(f)$ is the size of a minimal Boolean formula computing f and $k \approx 5.13$. Theorem 1.10.2(2), due to Wegener, then specializes Spira's result to monotonic f.

The final section of Chapter 1 is given over to a brief discussion of various other nonuniform models of computation, e.g., Boolean cellular automata, Hopfield nets, and so-called anonymous networks.

Chapter 2 (Circuit Lower Bounds). Given a Boolean function f of n arguments, one wishes to know the minimal size and minimal depth of circuits computing f. Since, by Immerman's Theorem 6.2.5, depth and size can be related to computation time and number of active processors, respectively, of parallel programs computing f, these issues have practical significance.

The work of Shannon and Lupanov settles the question of the size of fan-in-2 circuits of arbitrary depth in a certain sense. Namely, it is shown (Theorem 2.2.1) that, for almost all Boolean functions f of n arguments, the minimal size of a fan-in-2 Boolean circuit computing f exceeds $2^n/n$, which coincides with Shannon's upper bound from Chapter 1. In Corollary 2.2.1 it is shown that, for almost all monotone Boolean functions f, the minimal size of a fan-in-2 monotone Boolean circuit computing f exceeds $\frac{2^{n/2}}{10n}$. The latter result uses the notion of a *slice function* whereby all slice functions are monotone but not vice versa.

A classical result due to Nechiporuk gives a lower bound on the size of a Boolean formula computing function f in terms of the number of f's nonconstant *subfunctions*, where a subfunction of n-ary f is the $(n-k)$-ary function that is the result of holding k of its arguments constant.

Section 2.4 presents a proof, due to Haken and Cook, of an exponential lower bound on the size of monotonic real circuits of bounded fan-in computing the so-called Broken Mosquito Screen Problem. (Real circuits are composed of *real gates* representing unary or binary real-valued functions on the

reals.)

Sections 2.5 and 2.6 describe the application of the *random restriction method*, introduced initially by Furst, Saxe, and Sipser, in obtaining first a superpolynomial and then successively better exponential lower bounds for the parity function, which is important by virtue of the fact that many other Boolean functions can be expressed in terms of it. Important contributions due to Sipser, Håstad, and Razborov are covered here. In Section 2.7 algebraic lower bound techniques for obtaining circuit lower bounds for the majority and modulus functions, due to Razborov and Smolensky, respectively, are discussed. For example, the latter showed an exponential lower bound on the circuit complexity of function MOD_r even in the presence of MOD_p gates, where p is prime and r is not a power of p. (Boolean function MOD_p^n of n arguments yields output 0 just in case the sum of its arguments is a multiple of p.) Section 2.8 fills out the picture by presenting what is currently known concerning the computational power of MOD_m gates, where m is not assumed to be prime—results attributed to Barrington-Beigel-Rudich, and Tsai.

Characteristic of this genre of results concerning specific types of gates is Theorem 2.10.1, due to Beigel, showing that any Boolean function computable by a depth-d circuit having m majority gates is computable by a larger, depth-$(d+2)$ circuit having a single majority gate. Thus the number of majority gates can be reduced to 1 with little increase in circuit depth at least.

Chapter 3 (Circuit Upper Bounds). In this chapter, where many of the most interesting results are due to the authors themselves, ideas from permutation group theory are applied so as to obtain circuit upper bound results for families of Boolean functions, e.g., for families of characteristic functions of formal languages over alphabet $\{0,1\}$. To begin, the *invariance (automorphism) group Aut(f)* of n-ary function f is defined to be the set of permutations $\sigma \in S_n$ such that $f(x_1,\ldots,x_n) = f(x_{\sigma(1)},\ldots,x_{\sigma(n)})$. Further, a language L is said to *realize* a sequence $\langle G_n : n \geq 1\rangle$ of permutation groups $G_n \leq S_n$ if $Aut_n(L) = G_n$ for all n, where $Aut_n(L)$ is the invariance group of (the characteristic function of) $L_n = \{w \in L : length(w) = n\}$. Thus, as described in Exercise 3.11.1, the sequence $\langle D_n : n \geq 1\rangle$ of dihedral groups is realized by the language $(0^*1^*0^*)|(1^*0^*1^*)$, whereas Theorem 3.2.2 says in effect that the sequence $\langle A_n : n \geq 1\rangle$, where A_n is the alternating group on n letters, is not realized by any language L over alphabet $\{0,1\}$.

Various notions of representability, of permutation groups by Boolean functions, are introduced. Theorem 3.2.4 (Isomorphism Theorem) states that every $G \leq S_n$ is such that $G = Aut(f)$ for some "Boolean" $f : \{0,1\}^n \mapsto \{0,\ldots,\lfloor\log(n+1)\rfloor - 1\}$. (This amounts to G's being $\lfloor\log(n+1)\rfloor$-representable in the authors' sense.) The automorphism group of any undirected graph is 2-representable, and apparently that of any directed graph is 3-representable. (If self-loops are permitted, change 2 and 3 to 3 and 4, respectively.) The authors' Representation Theorem (Theorem 3.4.2) states that if $H = G \cap K$, where $H < G < S_n$ and K is k-representable for some k, then, among subgroups of G, we have that H acting on $\{0,1\}^n$ is maximal with regard to number of orbits—no permutation in G can be added to H without decreasing the number of orbits. It then turns out (Theorem 3.4.3) that $G \leq S_n$ is k-representable for some k just in case G is a subgroup of S_n maximal with respect to orbits. (Further, the least such k will satisfy $k \leq \binom{n}{\lfloor n/2\rfloor}$ and of course any k-representable permutation group is $(k+1)$-representable as well.) With just a "few" exceptions—these include A_n with $n \geq 3$—any maximal subgroup of S_n is 2-representable. (Here "maximal" is being used in the usual sense.) As shown by Kisielewicz, there exist 3-representable groups that are not 2-representable. However, it is not known apparently whether this holds for any $k > 2$.

Section 3.5 is given over entirely to a logspace algorithm, due to the authors themselves, that, given input (a disjoint-cycle representation of) a single permutation $\sigma \in S_n$, determines whether cyclic group $\langle\sigma\rangle \leq S_n$ is 2-representable and, if so, constructs a Boolean function representing $\langle\sigma\rangle$.

In §3.6 it is shown that almost all Boolean functions have trivial automorphism groups—just $\{id_n\}$—and this enables the authors to formulate and prove a certain 0–1 law for Boolean functions.

Let L(P) be the class of languages $L \subseteq \{0,1\}^*$ whose automorphism groups have polynomial index, that is, $|S_n : Aut_n(L)|$ is $n^{O(1)}$. In §3.7 it is first demonstrated that any L in L(P) is in nonuniform NC. This is then improved (Theorem 3.7.3) to L in TC^0 and hence in NC^1, where the former is the class of (Boolean functions computed by) constant-depth, polynomial-size circuits having unbounded fan-in threshold gates. (This discussion would be enhanced by some *examples* of languages in L(P).) In §3.8 the topic is the relation between the quantity of symmetry exhibited by language L (number of orbits of $Aut_n(L)$) and its computational complexity. The main result here is Theorem 3.8.3, due to Babai, Beals, and Takácsi–Nagy, stating that if the automorphism groups $Aut_n(L)$ of L are transitive with polynomially many orbits, then L is in TC^0.

The remainder of Chapter 3 is given over to a discussion of the bit complexity of anonymous networks, with either ring or hypercube topology, *qua* computers of Boolean functions. (By *bit complexity* we mean the total number of bits exchanged by all processors over an entire computation.)

Chapter 4 (Randomness and Satisfiability). The topic in this relatively short chapter is threshold phenomena for k-SAT: letting ϕ be an random instance involving m clauses (disjunctions) and n propositional variables, then it may happen that deciding satisfiability for ϕ is easy provided that clause-to-variable ratio $\frac{m}{n}$ is somewhat below ("easy yes") or somewhat above ("easy no") a certain threshold value. In §4.2 it is shown that in the case of 2-SAT this threshold value is 1. The case of 3-SAT is harder, which is hardly surprising given NP-completeness. Various techniques yield successively better upper bounds on an unsatisfiability threshold—higher ratios $\frac{m}{n}$ almost surely make for an easy no—that is seen to fall from 5.19 (Theorem 4.3.1) to 4.78 to 4.667 (Theorem 4.3.2) to 4.642 to an ultimate value of 4.601 (Theorem 4.3.3). (A threshold around 4.2 is predicted by empirical studies.) Section 4.4 takes up the other half of the story, first establishing a lower bound on a satisfiability threshold (easy yes below) for k-SAT generally and yielding $\frac{2}{3}$ for $k = 3$. Theorem 4.4.2, due to Frieze and Suen, improves this to 3.003.

The Satisfiability Problem is itself of interest, say the authors, due to its wide applications within artificial intelligence and software design. About this they are no doubt right. What is not clear, on the other hand, is what Chapter 4 has to do with the rest of the book, an issue to which we return below (see Opinion). At no point in Chapter 4 do the authors choose to refer—or need to refer apparently—to any other part of their text, and vice versa.

Chapter 5 (Propositional Proof Systems). At over 160 pages this is the longest chapter in the book by far. By a *propositional proof system* one means a polynomial-time-computable onto function $f : \Sigma^* \mapsto \text{TAUT}$, where Σ is some finite alphabet. Such an f is *polynomially bounded* provided that there exists a polynomial p in a single variable such that every member $\mathcal{A}$ of TAUT is the image of a "proof" in Σ^* whose length does not exceed $p(|\mathcal{A}|)$. As shown by Cook and Reckhow in their 1977 paper, there exists a polynomially bounded propositional proof system for TAUT just in case NP =? co-NP (Theorem 5.2.1). This result has given rise to a program of establishing lower bounds on sizes of proofs, within natural propositional proof systems, of a variety of combinatorial principles, all in an effort to relate the existence of polynomially bounded proof systems to P =? NP. The principles considered include the various versions of the Pigeonhole Principle (PHP) that the authors present on page 254. A notion of *polynomial simulation* of one proof system by another facilitates comparisons of distinct proof systems with respect to proof strength. In brief, a propositional proof system $\mathcal{P}_1$ *simulates* $\mathcal{P}_2$ just in case, for any proof Q of tautology $\mathcal{A}$ in $\mathcal{P}_2$, there is a proof P of $\mathcal{A}$ in $\mathcal{P}_1$, where the length of P is polynomial in the length of Q.

In successive sections of Chapter 5, the authors consider the following proof systems and their several variants: the Gentzen sequent calculus LK, Robinson's resolution refutation, various algebraic refutation systems (*Nullstellensatz* and the polynomial, Gaussian, and binomial calculi), the cutting plane proof system, and Frege systems. A characteristic result (Theorem 5.3.5) states that, although there exist polynomial-size LK proofs, using cut, of a certain version of PHP, the length of any cut-free proof has superpolynomial size. In §5.8 eleven open problems are described in some detail.

Chapter 6 (Machine Models and Function Algebras). This chapter consists largely of the first author's contribution to E. Griffor, ed., *Handbook of Computability Theory* (Elsevier, 1998). By a *function algebra* one means the class of functions obtained by first introducing certain "initial" functions and then closing under various function-forming operations. The study of the relationship between machine models of computation and function algebras is motivated by an interest in presenting machine-independent characterizations of complexity classes, typically defined in terms of particular machine models, in terms of function classes. (The authors suggest that the latter are to the former as software to hardware.) This chapter is a survey of techniques used to characterize complexity classes using function algebras.

Among sequential models of computation, the authors emphasize alternating multitape Turing machines (ATMs). Complexity class ALOGTIME is defined as the class of languages accepted by ATMs in time logarithmic in the length of input. Restricting ATM computations to some fixed constant number of alternations leads to the definition of the *logtime hierarchy* LH. Two results concerning ALOGTIME are highlighted. First, the authors present Barrington's proof that a certain word problem for any given finite nonsolvable permutation group is complete for ALOGTIME under DLOGTIME reductions. Second, they mention S. Buss' result showing that a certain "Boolean formula valuation problem" is similarly complete.

As for parallelism, the *concurrent random access machine (CRAM)* model is described in §6.2.2. A CRAM is a finite sequence $R_1, \ldots, R_m$ of random access machines, each with its own local memory and an infinite collection of registers each capable of holding a single natural number. In addition, a CRAM possesses an infinite collection of global registers $M_0^g, M_1^g, \ldots$ accessible to all R_i and used for reading input, passing messages between processors, and writing output. Global M_i^g may be read concurrently by several processors, and *priority resolution* is used to handle write conflicts. While operating synchronously on the same program, the various R_i may have different data in their respective local memories. Further, they recognize a unique processor identification number (PID) so that one and the same instruction ("increment $M_{\mathrm{PID}}^g \ldots$") changes in meaning over the various R_i. The CRAM instruction set includes local operations, global operations for reading and writing, and control instructions ([conditional] GOTOs and HALT). A sampling of parallel algorithms, focused on string matching, are presented following unpublished notes of E. Kaltofen.

Uniform circuit families, whereby some uniformity criteria relate the members of $\{C_n : n \in \mathbf{N}\}$ to one another, are investigated in Chapter 6. Uniform AC^k, i.e., those Boolean functions whose n-ary restrictions are computed by LOGTIME-uniform circuit families, is of particular interest: these functions f are precisely those whose n-ary restrictions f_n are computable in $O(\log^k n)$ computation steps by a CRAM with polynomially many active processors—Theorem 6.2.5 mentioned earlier.

In §6.3 various recursion schemes are surveyed, and the resulting function algebras are then related to complexity classes derived from the ATM and CRAM models primarily. Application of Gödel arithmetization techniques to machine model $\mathcal{M}$ presupposes that a corresponding function algebra contain string manipulation functions and the like sufficient to ensure the availability of functions NEXT_M such that $\mathrm{NEXT}_M(x, c) = d$ whenever c and d encode successive configurations of machine M for input x, where M is an instance of $\mathcal{M}$.

A typical result here is the description, due to the first author, of a function algebra A_0 for the class of number-theoretic functions with polynomial growth (whose bit-graphs) are in LH (Theorem 6.3.3). (A k-ary number-theoretic function f *has polynomial growth* provided that $|f(x_1, \ldots, x_k)|$ is $O(max_{1 \leq i \leq k} |x_i|)$ for some k, where $|x| = \lceil \log_2(x+1) \rceil$.) To form A_0 one first introduces the following initial functions: the constant-0 function, k-ary projection functions for arbitrary k, binary successor functions s_0 and s_1 defined by $s_0(x) = 2 \cdot x$ and $s_1(x) = 2 \cdot x + 1$, length function $|x|$ defined above, binary function $\mathrm{BIT}(x, i)$ returning the i^{th} bit within the binary representation of x, and the binary *smash function* defined by $x \# y = 2^{|x| \cdot |y|}$. One then closes under function composition and so-called *concatenation recursion on notation*. It then follows, by an argument involving AC^0-reducibility, that TC^0 is identical with the function algebra formed like A_0 but with the additional inclusion of integer multiplication among the initial functions (Theorem 6.3.5 due to Clote and Takeuti).

Chapter 7 (Higher Types). By a *type-2 functional* one means a total mapping taking $(k+\ell)$-tuples consisting of k number-theoretic functions and ℓ natural numbers to a single natural number. The associated sequential computation model is that of the *oracle Turing machine (OTM)* that, in addition to the usual input, output, and worktapes, possesses an oracle query tape and oracle answer tape for each function input. (There is also an oracle query state for each function input.) The collection *BFF* of *basic feasible type-2 functionals* is defined, and a result of Kapron and Cook relating them to OTMs is cited (Theorem 7.2.2). (The proof is left to Exercise 7.8.5.)

Analogous to the collection A_0 of functions that played a central role in Chapter 6, the collection $\mathcal{A}_0$ of type-2 functionals is the smallest class containing certain initial functionals and closed under certain functional-forming operations (Definition 7.3.3). Most of Chapter 7 is given over to showing that type-2 functional $F(f_1, \ldots, f_k, x_1, \ldots, x_\ell)$ is in $\mathcal{A}_0$ just in case it is computable by an *oracle concurrent random access machine (OCRAM)* M_F in constant time, where the number of M_F's processors is polynomially related to arguments $f_1, \ldots, f_k, x_1, \ldots, x_\ell$ (Theorem 7.5.3). The OCRAM model, which permits simultaneous oracle calls to input functions by its several processors, is described in §7.5, and several longer examples of OCRAM programs are given.

Opinion. Although some the material covered in *Boolean Functions and Computation Models* can be found elsewhere in the secondary literature—we are thinking particularly of circuit lower bounds (Chapter 2) and propositional proof systems (Chapter 5)—there is no other text that brings together such a wealth of diverse results concerning the research topics at issue. The very interesting material in Chapter 3, on the other hand, is found nowhere else to our knowledge. Although we did find the discussion in Chapter 2 of Sipser's random restriction method—the lead-in page 91 in particular—hard to follow, the authors' exposition is usually quite clear, and whatever small deficiencies exist in that regard can surely be overlooked given the book's prodigious scope and depth. (The descriptions of the various propositional proof systems in Chapter 5 and of the several computation models in Chapters 6 and 7 are especially nice.) In their preface, the authors suggest that the individual chapters can be read independently, and this really does seem to be the case, since definitions of terminology and notation are often repeated as needed.

In the case of almost every theorem, a full proof is given and its source cited. Various pseudocodes for presenting algorithms are introduced and adhered to consistently at least within any single chapter. The final section of each chapter is devoted to historical and bibliographical remarks. There are ten to forty exercises, of various degrees of difficulty at the end of each chapter. These occasionally include open problems. Finally, there is an extensive bibliography, and the index, mixing symbols and terms, is perhaps adequate for most readers. (There is no name index.)

For anyone wishing to get quickly to the forefront of current research with respect to the topics it covers, this book is doubtless a good place to start. On the other hand, the book's tremendous

scope means that the demands on the reader are considerable. For instance, the reader will find only two examples, with diagrams, of Boolean circuits—on pages 8 and 436—and they are essentially the same.

Our one reservation concerns the organization of the book across chapters: specifically, the text gives the impression of being, in a sense, not one book but, rather, four. The first extends over Chapters 1 through 3, covering Boolean functions and complexity classes derived from the Boolean circuit model. In later chapters these ideas do occasionally resurface to be sure. For example, the combinatorial argument for a certain Boolean circuit lower-bound result from Chapter 2 figures in an argument for an exponential lower bound on the size of proofs of a version of PHP within bounded-depth Frege systems. But such connections typically receive no mention within chapter introductions and are, for the most part, buried within the text. This has the unfortunate consequence that Chapter 5, say, on propositional proof systems will probably strike most readers as largely unrelated to any of the other chapters, although Pudlák's Theorem 5.6.7, relating cutting planes refutations to real circuits, and remarks on pages viii and 366 point in the right direction. Chapter 4 is stand-alone, as mentioned earlier. Finally, Chapters 6 and 7, where complexity classes derived from the ATM and CRAM (OCRAM) models are characterized by function(al) algebras, also stand as a unit. Ultimately, the results mentioned at the end of our discussion of Chapter 6 do make for connections to complexity classes derived from the Boolean circuit model and introduced in Chapter 1. And Immerman's Theorem 6.25 is no doubt important in this regard. But these links are not cited in the authors' brief overview on page 415, and the overall impression, once again, is that of disjointness.

Perhaps the authors chose not to emphasize interconnections, between the several research areas covered, just because, as they stand, these interconnections seem not so impressive. And this situation no doubt reflects, to some extent, the current state of knowledge. The challenge implicit in *Boolean Functions and Computation Models* is that of finding interesting ways to relate Boolean circuit families to propositional proof systems, say, so that it is more often the case that lower bounds on depth for the former yield lower bounds on proof size within the latter. And it is likely that the appearance of this stimulating book will lead its readers in just such directions.

Review of[3]
Selected Papers in Discrete Mathematics
by D. Knuth
Published by CSLI (Center for the Study of Language and Information Publication
Paperback, 286 pages, $72.00
$42.00 on Amazon used

Review by Carlos A.S. Oliveira
Dept. of Ind. & Systems Engineering, University of Florida

1 Introduction

The book "Selected Papers in Discrete Mathematics" is a compilation of work published (plus some unpublished papers) by Donald E. Knuth, in the area of discrete mathematics. Clearly, the author does not need any introduction, and is well known by his authoritative work in areas such analysis of algorithms and digital typography. However, more than this, Knuth is a great example of good

[3]copyright 2004, Carlos Oliverira

expositor and writer. Thus, even if your interests are not directly related to the more mathematical areas of computer science, it is greatly rewarding to read this book as a way of learning how to write good papers.

The selection and ordering of the papers in this volume seems to have been performed more by subject than chronological order. Although there is not a clear indication of such a division (which can be seen as a minor organizational issue), one can identify from the table of contents the major topics that have been addressed in the papers forming the book. Initially, for example, there are a couple of papers related to the issue of notation, then exposition papers on history of discrete mathematics. Other than the table of contents, a good way to find topics covered in the book is checking the comprehensive index.

2 Topics Covered

The book is extensive, with 812 pages, index included, of mostly mathematical topics. Thus, I will not make a review of every paper, since this would be boring to readers and probably not worthwhile the effort (one can check the preface for a brief description of the contents of each chapter). I will instead describe the main areas that are covered by the book and provide examples based on the papers on each area. If you have interest, keep in mind that there is much more that was not discussed here.

The first chapter of the book describes how computers can be useful as a tool for helping the development of mathematics. The paper, written in 1965, serves roughly as a statement of the type of mathematics the author is interested in. The use of computers in discrete mathematics has only increased since the time the paper was written, thus it is interesting to compare the results stated there with what we have today.

The first major topic discussed in the book is related to the use of notation in discrete mathematics, and how they can influence the discovery and development of useful results. One of the interests of Knuth has been the establishment of good mathematical notation, especially to be used in his series "The Art of Computer Programming." Some of the notations discussed in these papers are:

- the so called *Iverson's convention*, where square brackets around an expression are used to return a value 1 if and only if the expression is true. This can be useful to simplify summations and other types of formulas used in discrete mathematics;
- a different convention for writing Stirling numbers, where a Stirling number of the first type is written as $\begin{bmatrix} a \\ b \end{bmatrix}$, and of the second type as $\begin{Bmatrix} a \\ b \end{Bmatrix}$. This leads to much simplification in many of the formulas including Stirling numbers, and new insights as well. For example, the simple result
$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} -b \\ -a \end{bmatrix}$$
can be better appreciated only due to this improved notation, as explained in the paper;
- the notation $[F(z)]G(z)$ for the vector product of two polynomial functions is shown to lead to easier manipulation of this type of operations in Chapter 3.

The next topic discussed in the book is history of discrete mathematics. There it is included an interesting paper about the work of Johann Faulhaber on summations of powers. As the paper

describes, it is not exactly known what method he used (at the time mathematicians where more interested in results than in proofs), but he accomplished amazing computational achievements for the time. A puzzle proposed by Faulhaber is solved using computers, showing that he did correct computations for very large numbers considering the time (he died in 1635). Another paper (Chapter 5) discusses the work of Thomas Harriot, who did impressive discoveries in the area of discrete mathematics in the early 17th century. Another paper of historical interest is Chapter 18, since it was the first paper published by Knuth in discrete mathematics. It presents amusing work on number representation, despite some inaccuracies that are pointed out in the postscript. (This shows that, by the way, even expert researchers can make mistakes early in their careers.)

Matrices are important objects in discrete mathematics. They are constantly used as a representation means as well as main objects of study. Many chapters in the book are related to matrices. For example, Chapter 6 describes a very interesting result by Egorychev (1980), proving a conjecture of van der Waerden (made in 1926) about the permanent of doubly stochastic matrices. The concept of *Pfaffian* of a matrix, a generalization of determinant, is discussed in the next chapter, together with a history of the discovery and use of Pfaffians. *Combinatorial matrices* are the subject of Chapter 9. A combinatorial matrix is a type of matrix expressed in general as the combination $bJ + (a - b)I$, where a and b are scalars, J is the matrix of ones, and I is the identity matrix. The chapter introduces some properties of this type of matrix.

Graph theory is another fundamental topic in discrete mathematics, thoroughly explored in this book. Topics such as spanning trees and graph enumeration are discussed in connection with diverse problems. A first paper (Chapter 8) presents the well know *theta function*, introduced by Lovász as a graph parameter whose value is always between the clique number and the chromatic number of a graph. The paper is a nicely written introduction and survey of the concept of theta function and its related characterizations. Chapter 10 describes properties of the so called *Aztec diamond* graphs, such as the number of spanning trees in such graphs. The spectra of matrices representing the tensor product of graphs is discussed in Chapter 11. The number of subtrees of a digraph is the topic of Chapter 12. A generalization of a theorem of Cayley (1889), giving the number of directed trees in a graph, is described in the next chapter (13). The generalization involves graphs with an associated coloring of nodes, such that some rules concerning the orientation and the colors adjacent to arcs must be enforced. The proof amounts to defining a correspondence between graphs and integer sequences and using this correspondence to calculate the number of directed trees. Later in the book (Chapter 24) interesting work is presented on the representation of strongly connected directed graphs.

The next major topic covered in the book is manipulation of polynomials. A computationally oriented paper appears in Chapter 15, showing how to use properties of convolution polynomials using the *Mathematica* package. "Polynomials involving the floor function" is a paper devoted to identities and general properties of polynomials, where variables are modified using the floor operator. Numerous applications in computer science use the floor function, and therefore it is important to understand how this operation can be manipulated when used on polynomials.

Discrete mathematics also plays an integral role (no pun intended) in modern algebra, for example in the study of finite fields and groups. Work in this area is presented in some of the chapters. Topics such as finite fields (Chapters 19 and 20), projective planes over semifields (Chapters 20 and 21), and grupoids (Chapter 22) are treated in full depth.

Recurrence relations in general appear on Chapters 37 through 39. In the first paper, linear recurrences with constant coefficients are discussed. In the second paper, a recurrence of the type

$$M(n) = \begin{cases} g(0) & \text{if } n = 0 \\ g(n) + \min_{0 \le k < n}(\alpha M(k) + \beta M(n-k)) & \text{for } n > 0 \end{cases}$$

is shown to be convex in some cases, being efficiently computable. In the third paper, a recurrence related to trees in random graphs is studied.

The last two papers are a real "tour de force", discussing ideas from the important area of random graphs. This area is concerned with the properties of graphs whose edges are selected randomly, according to some distribution, and was championed by Erdös and Rényi. Some of the basic results in random graphs state that, after a specific threshold is achieved by the probability distribution function, most graphs will have some specific property such as connectivity, for example. The second last paper in the book describes the properties of initial cycles formed in a evolving random graph (i.e., a graph in which edges are being added according to a probability distribution). For example, one such property of interest is the distribution of cycle lengths. The last paper, which is the largest in the book (150 pages, filling up a whole issue of the *Random Structures and Algorithms* journal) describes results related to one of the most fascinating phenomena in random graphs: the appearance of a giant component during the random process of adding edges.

Other topics appear in different parts of the book. For example, matroids (Chapters 26 and 27), permutations and partitions (Chapters 28 and 30 to 35), coding theory (Chapter 29), and number theory (Chapter 36).

3 Conclusion

As seen above, "Selected Papers in Discrete Mathematics" is a book that covers a long range of material of interest to people working in combinatorics, theoretical computer science, optimization, and other related areas. However, it must be said that although some of the papers cover basic concepts, the book is not suitable for a beginner to learn any of these topics. There are too many, sometimes very distant topics being discussed, such as random graphs and algebra.

I would say that the audience for the book comprises two groups. First, researchers working with discrete mathematics, which may want to have a good collection of papers in one of the areas covered. A second group that would benefit from the book are professionals interested in how to write good papers, and willing to learn at least a little of mathematics.

This said, the book is well deserving the price. The content is written in an authoritative way. Suffices to say that these papers have been published in some of the best journals in the area, and they were written and throughly revised by the same author of "The Art of Computer Programming."

Review of[4]

Linear Optimization and Extensions – Problems and Solutions
by Dimitris Alevas and Manfred Padberg
Springer-Verlag, 450 pages, $54.95, Softcover

1 Introduction

Linear Optimization and Extensions – Problems and Solutions – is a solution manual for another book, released in 2000, by the second author [1]. The original book is a comprehensive introduction to the topics of linear programming and combinatorial optimization. The current book, therefore, has the initial goal of providing hints, and solutions for the problems there stated.

[4]copyright 2004, Carlos Oliveira

However, the authors managed to create a book that is far from being a simple solution manual. This is clearly expressed in the introduction, where they describe the development process, as well as their final objectives. They decided to write a self contained text, with summaries of the most important points in each chapter, followed by solved exercises. Also, some of the presented exercises are new, and were not included in [1]. The result is a well-written reference text, with much more material than what can be expected of a solutions manual.

Each section of the book starts with a summary of the important facts appearing the corresponding sections of [1]. All statements in the exercises are included, thus the student does not need a copy of [1] to understand the problems. These features allow the book to be used as a self contained reference.

2 Book Content

Introduction In the introduction, the topic of LP modeling is covered, with the help of several examples. The examples, here called mini-cases, correspond to standard situations in industry where LP theory is helpful. The examples are discussed in depth, and completely solved with employment of computer packages, such as cplex and matlab. This chapter is directly relevant for students who want to apply LP in the industry, with complete coverage of the LP modeling cycle, and description of solving tools. However, it seams a little far fetched and even boring for students interested only in the theory.

Linear Programming The second chapter deals with basic definitions of linear programming. The standard formulation

$$\min cx \qquad \text{such that } Ax = b,\ x \geq 0$$

is presented, together with the related canonical form

$$\max cx \qquad \text{such that } Ax \leq b,\ x \geq 0.$$

A basic result of linear programming is that these two forms are equivalent, what can be shown using simple linear transformations. The chapter also presents other fundamentals from linear algebra, which are very important to the clear understanding of linear programming algorithms.

Basic Concepts In this chapter the main operations on linear programs are defined and exemplified. Concepts such as *base*, *feasible solution*, *degenerate solution* are defined, as well as the geometric representation of a linear program, in terms of coordinates in the R^n. The exercises reinforce the definitions and examples, in an effort to help students develop the intuition of polyhedral geometry.

Five Preliminaries This chapter introduces the basic concepts of the simplex method. The five preliminaries needed to understand the simplex method are: *basic feasible solutions*, which are an algebraic representation for extreme points, the geometric points where optimal solutions are located; the *sufficient optimality criterion*, which says when a solution is optimal; the *unboundedness criterion*, which determines if a problem has unbounded solution; the *rank-one update*, used to find the inverse of a matrix after a multiplication by the rank one matrix uv^T; and finally, the *basis change*; which is the method employed to change from a basic feasible solution to another.

Simplex Algorithms In chapter five the simplex algorithm is introduced. Several variants of the basic algorithm are discussed, with different pivot column and row selection mechanisms. The starting phase of the simplex is presented, using the Big-M and the two-phase methods. The exercises in this chapter emphasize the algorithmic aspects of the simplex method. In particular, sample implementations of the algorithms discussed are shown, using matlab. This alone is an interesting feature, since most text books do not care about real implementation issues. The exercises in this chapter serve also to formalize many of the results, which are necessary to prove the correctness of the simplex, but can become boring if introduced in the initial explanation.

Primal-Dual Pairs Duality is one of the main properties of linear programming, and one can say that this is the reason why LP is so popular as a tool for developing and proving the correctness of algorithms. For example, in the theory of approximation algorithms, duality is frequently used to prove approximation properties. As another example, in graph theory LP duality can be used to prove the classical maximum flow/minimum cut theorem. Therefore, duality it is a property useful not only for the simplex method, but for all algorithms that work with linear models. In this chapter, the authors describe initially the mechanics of dual problem construction. The strong duality theorem is then introduced, which leads to the very important *complementary slackness* conditions: if x is the optimum solution for the primal problem, and u is the optimum solution for the dual, then

$$u(Ax - b) = 0 \quad \text{and} \quad (uA - c)x = 0.$$

The duality theorem is used to prove the Farkas' Lemma, a fundamental result that can be used to build duality theory, being also useful in many other areas. The exercises in this chapter explore important topics such as the dual simplex, again with sample implementation.

Analytical Geometry Chapter seven presents what seams to be a main departure from the traditional method used to teach linear programming. Building on their experience in the field of polyhedral techniques, the authors present an introduction to Analytical Geometry. This is a prerequisite to the proper understanding of much of the modern algorithms for linear programming, as well as its generalizations. However, in most books the material related to this topic is kept to a minimum. The information contained in this chapter is also very useful for a complete understanding of integer programming techniques explained latter. This is one of the features of this book, and can become quite interesting for readers who want to explore other texts such as "Integer and Combinatorial Optimization", by Nemhauser and Wolsey.

Projective Algorithms In the eighth chapter, the interior points method is explained. This is a class of polynomial algorithms for LP developed in the last two decades, which provides the best performance for some large linear programs. The ideas used in its development use many of the geometrical concepts described in chapter seven. The presentation in the book gives a large number of geometrical insights, which help in the clear understanding of the technique. Some extra topics, such as the logarithmic barrier function, which is very useful in current implementations of LP solvers, are also explained in the introductory section.

Ellipsoid Algorithms In chapter nine, the ellipsoid algorithm for linear programming is introduced. This algorithm has enormous importance in the theory of linear programming. Initially, because this was the first polynomial time algorithm to solve linear programming problems (the existing methods had exponential cases, despite the great efficiency on average of the simplex

method). However, the ellipsoid method is also important in proving results about problems other then linear programming, due to its generality. One interesting result derived from the ellipsoid algorithm is the celebrated equivalence between optimization and separation; that is, the complexity of separating an infeasible point from the feasible set (using a plane), is the same as that of the optimization problem. The book provides a very understandable summary of the facts concerning the ellipsoid method, again with a sample computational implementation (which is not very efficient, due to problems with the algorithm itself).

Combinatorial Optimization Chapter ten gives an introduction to one of the most important applications of linear programming theory: combinatorial optimization. Such problems have the characteristic that some of their variables have integer value, which makes them much harder to solve. The general *mixed-integer linear* program can be described as

$$\max\{c\vec{x} + d\vec{y} \mid A\vec{x} + D\vec{y} \geq \vec{b}, \vec{x} \geq \vec{0}, \vec{y} \geq \vec{0}, \vec{x} \in \mathbb{Z}^n, \vec{y} \in \mathbb{R}^m\}.$$

Due to results in computational complexity, it is believed that there is no efficient algorithm to solve such problems. Thus, the basic solution method uses *implicit enumeration* techniques, such as *branch-and- bound*. This approach is described in the introduction, and developed in exercises, together with extensions of the basic technique, such as the *cutting plane methods*. This chapter also explain the notions of *complete* and *ideal* formulations, used to describe a minimal polyhedron containing the feasible solutions to a combinatorial optimization problem.

Appendices Some extended applications are described in the appendices. Particularly interesting is the description of problems on printed circuit boards, used to introduce the traveling salesman problem. Some of the most important techniques developed in text are applied to solve the examples. This constitutes a nice explanation of the real usage for the presented theory.

3 Conclusion

The book is a very well planned and written. It presents a large number of exercises, examples, programming codes, and interesting applications of the theory of linear programming. Having said this, it must be clear that it is not meant to be a complete textbook, but just a complement to existing texts. One of the advantages of giving introductory notes in each section is that, for students who already know the theory, this can become a quick reference for the main results. For professors and interested students, the book can also serve as a source of advanced exercises.

An important feature of the book, which sets it apart from other introductory texts, is the inclusion of details about analytical geometry and projective algorithms. This can be confusing to first time readers, but for people really interested in knowing the internal mechanisms used by most modern algorithms, this is certainly the way to go. Also, the publication of sample computer code is interesting to many people, and a feature not found in most books about LP.

In conclusion, I believe that the book will prove invaluable not only for students, but also for professionals in industry and universities. This is certainly an important addition to the literature of the area of linear programming.

References

[1] Manfred Padberg. *Linear Optimization and Extensions*, volume 12 of *Algorithms and Combinatorics*. Springer Verlag, 2nd edition, 2000.

Review of[5]

Introduction to the Design and Analysis of Algorithms
Author of book: Ananay Levitin
Hardcover, 528 pages, Addison-Wesley, $85.00
On Amazon Used its just $32.00

Reviewer: William Fahle

This is a beginning algorithms book, suitable for the late undergraduate or early graduate computer science student. Subjects covered include an introduction of the concept of algorithms, a taxonomy of various algorithms and design techniques, up through a preliminary discussion of the topics of complexity and computability. The subject of algorithms is, of course, well known and fundamental to the computer sciences. This book provides an introduction to the subject through the use of some well-known and not so well-known puzzles and computing problems.

The author endeavors in this work to establish a new taxonomy on algorithms. Rather than group algorithms by the type of problem they solve such as sorting or searching, he classifies them by the approach which is taken to solve the problem. Furthermore, the classic division of approaches into divide-and-conquer, greedy, and dynamic programming is refined and extended to include many common techniques. The general techniques include brute force, decrease-and-conquer, transform-and-conquer, and even some techniques for NP-hard problems such as branch-and-bound and backtracking.

Computer-based problems such as sorting, searching, string processing, and graph problems are introduced, and puzzles such as the Knigsberg bridge puzzle posed by Euler are used as examples for these problem types. The puzzles allow us to see the nature of problems that are easy to describe but perhaps difficult to solve.

Essential data structures are introduced, and the formal notation for performance analysis is described in detail, including big-oh, big-omega, and big-theta and what they mean mathematically. The descriptions are very accessible, yet more advanced concepts such as the master theorem and recurrence relations are also covered.

With these preliminaries out of the way, the text proceeds to introduce design techniques one chapter at a time. Brute force is shown first, since this is often the obvious solution to any given problem. By contrast the remaining approaches are shown to be more efficient if it is possible to do so; problems for which no known better solution is known are also treated.

The next chapter covers divide-and-conquer, which is perhaps one of the easiest techniques to understand for the beginning student. Next come decrease-and-conquer and transform-and-conquer. The latter refers to simply solving a problem for a smaller input (usually recursively) then using the result to solve the whole problem. The former refers to techniques such as reduction or pre-sorting.

Chapter seven covers space and time tradeoffs, which arguably is not a design technique, but there are several algorithms that exist entirely on the basis of such tradeoffs. These include b-trees, hashing, and Boyer-Moore string matching. The next chapter is about dynamic programming, which is perhaps a generalization of space-time tradeoffs, but problems in this domain tend to have an optimal sub-problem which can be used to solve the larger problem.

The chapter on dynamic programming is followed by a section on the limitations of algorithmic power and methods of coping with those limitations, such as approximation algorithms and schemes. There is then an epilogue in which the material of the book is summarized, followed by a couple of good appendices on the mathematics of algorithm analysis and recurrence relations, along with

exercise hints. The bibliography is extensive, and the eighteen-page tagged index is comprehensive; algorithms are listed by name and by type. Most of the games and puzzles used in the text are indexed, along with special notation and numbers.

Rather than give solutions to some of the exercises, the author chose to give hints for every problem. These hints will hopefully stimulate further thinking about each exercise, and certainly the student should attempt the problem before looking at the hint.

Overall, this book, while not revolutionary, certainly has many points to recommend it. The author has written in an accessible style with a solid aim to educate the reader rather than obscure the content in needlessly complex language or notation. Nevertheless, the content is rigorous and complete as far as it goes, and the puzzles used in examples along with the Socratic injection of questions to the reader along the way make the read enjoyable and worthy of attention.