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. **Classical and Quantum Computing with C++ and Java Simulations** by Yorick Hardy and Willi–Hans Steeb. Reviewed by Nick Papanikolaou, 2005. This is an undergraduate textbook which, as the title promises, has a rather wide coverage.

2. **Integer Programming** By Laurence A. Wolsey. Reviewed by E. Jonathan Chapin. This is a book on Integer Programming that is aimed at Advanced Undergraduates.

3. **Computational Line Geometry** by H. Pottmann and J. Wallner. Reviewed by Georg Essl. This is *not* a book on Computational Geometry, and it is *not* a book on Numerical Line Geometry. Its a new look at an old field— line geometry looked at in a way that can be used for computation.

4. **Logic for Computer Scientists** by Uwe Schöning. Review by Riccardo Pucella. This is a short introductory book on propositional and first-order logic, with a bias towards computer scientists. This is, its emphasis is computational.

5. **Teaching Statistics Using Baseball** by Jim Albert. Review by James Glenn. This is an undergrad text in statistics that uses Baseball for its main examples.

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. *Combinatorial Optimization: Packing and Covering* by Cornuejols.

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

Books on Complexity, Cryptography, and Combinatorics

1. *Cryptography in C and C++* by Welschenbach

2. *Complexity of Classification of Boolean Constraint Satisfaction Problems* by Creignou, Khanna, and Sudan.

3. *Elliptic Curves: Number Theory and Cryptography* by Larry Washington.

4. *Coding for Data and Computer Communications* by Salomon.

5. *Coding Theory: A first course* by Ling and Xing.

[1]© William Gasarch, 2005.

6. *Block Error-Correcting Codes: A Computational Primer* by Xambo-Descamps.

7. *Combinatorial Designs: Constructions and Analysis* by Stinson.

Misc Books

1. *The Elements of Computing Systems: Building a Modern Computer from First Principles* by Nisan and Schocken.

2. *Alan Turing: Life and Legacy of a Great Thinker* Edited by Teusher.

3. *Ordered Sets: An Introduction* by Bernd Schroder.

4. *Algorithmic Learning in a Random World* by Vovk, Gammerman, Shafer.

5. *Domain Decomposition Methods– Algorithms and Theory* by Toseli and Widlund.

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

7. *Semantic Integration of Heterogenous Software Specifications* by Martin Groβe-Rhode.

8. *Handbook of Computational Methods of Integration* by Kyther and Schaferkotter.

Review of[2]
Classical and Quantum Computing
with C++ and Java Simulations
Authors: Yorick Hardy and Willi–Hans Steeb
Publisher: Birkhäuser Verlag, 2001
$77.95, Paperback, 589 Pages

Reviewer: Nick Papanikolaou
(Dept. of Computer Science,
University of Warwick, U.K.)

1 Introduction

This textbook is an almost encyclopædic tome on a range of topics mainly within the undergraduate computer science curriculum. Its coverage is very broad, starting from boolean algebra and proceeding all the way to implementations of quantum computers. The selection and ordering of topics is somewhat arbitrary, and a book could be written about the subject matter of nearly every chapter. The most distinctive feature of this text is its coverage of quantum computing. However, while the title may suggest a balanced treatment of classical and quantum computing, only 30% of the book's pages are devoted to the latter. One may wonder what the intended readership is, since there are several good texts devoted to quantum computing, and many many more on the topics covered in the other 400 pages.

[2]© Nick Papanikolaou, 2005

2 Coverage

The book is divided into two parts; Part I ('Classical Computing') consists of 15 chapters. Part II, 'Quantum Computing,' consists of 8, generally shorter, chapters. What follows is a summary of the topics covered.

Chapter 1: Algorithms. The first chapter introduces some basic terminology used in the analysis of algorithms, as well as mathematical induction. Euclid's algorithm is used as an example of a classical algorithm; the process of *simulated annealing* is used as an example of a randomized algorithm. Some simple programs in C++ and Java are given.

Chapters 2–7: Boolean Algebra, Sequential and Combinational Circuits. Chapter 2 presents the laws of boolean algebra, De Morgan's theorem, Karnaugh maps and the Quine–McCluskey method for circuit simplification. This is followed by a chapter on number representation, discussing implementations of integers and real numbers. The next two chapters present examples of boolean circuits such as adders and programmable gate arrays, as well as latches. Synchronous circuits take up a chapter of their own.

Chapter 8: Recursion. This chapter discusses recursion in general, and illustrates the technique with examples (Towers of Hanoi, integer powers, quicksort and more) and sample programs. The wavelet transform is discussed; backtracking is illustrated through the 8–queens problem. Implementation of recursion with and without stacks is also covered.

Chapter 9: Abstract Data Types. The book presents the linked list, stack and tree ADTs in a single chapter. Most of the chapter consists of C++ code which implements these ADTs. The explanation of abstract data types is particularly brief, and is only useful as a reference for programmers.

Chapter 10: Error Correction and Detection. The tenth chapter treats Hamming codes and Shannon's theorem. Some useful C++ code for calculating Hamming codes is given, and there is also a Java program for computing weighted checksums.

Chapter 11: Cryptography. The chapter on cryptography is oriented towards programmers, with sample code for transposition and substitution ciphers and the RSA cryptosystem. Relevant background material is discussed only briefly.

Chapter 12: Finite State Machines. Chapter 12 covers finite automata; apart from general background material, attention is paid to Moore machines, Mealy machines and Turing machines. A C++ listing of a simulator for a Turing machine is given.

Chapter 13: Computability. In a mere 10 pages, the authors describe Church's thesis, Gödel's theorem and NP–completeness.

Chapter 14: Neural Networks. The fourteenth chapter is a good, readable presentation of neural networks, detailing the McCulloch–Pitts neuron, the perceptron and its learning algorithm, the multilayer perceptron and back–propagation. Coverage of neural networks is significantly more detailed than that of material in previous chapters. Sample code is provided for simulating networks and back–propagation.

Chapter 15: Genetic Algorithms. This chapter is one of the stronger points of the book, as it deals with an area of considerable academic interest. It discusses the basic, sequential genetic algorithm, the Gray code, Markov chain analysis, and maxima of one- and two–dimensional maps. C++ and Java code for genetic algorithms and related applications (e.g. knapsack and traveling salesman problems) is given.

Part II of the book starts with Chapter 16, which introduces the mathematics of quantum theory.

Chapters 17–18: Basics of Quantum Computing and Quantum Measurement. Both of these chapters tackle fundamental aspects of quantum computation and quantum information. Qubits and quantum registers are introduced, and so are quantum gates and circuits. Quantum measurement is discussed in a chapter of its own, along with certain interpretations (e.g. the Copenhagen interpretation, Everett's many worlds interpretation) of the process. SymbolicC++ code for simulations of quantum circuits are provided, and are generally useful and helpful for practically oriented computer scientists.

Chapters 19–20: Quantum State Machines and Teleportation. By analogy to the material in the chapter on finite state machines, the authors have devoted a few pages (Chapter 19) to quantum automata and quantum Turing machines. The quantum teleportation protocol, which allows the transfer of a single qubit using only a classical channel and a pair of entangled particles, is covered in a short chapter.

Chapter 21: Quantum Algorithms. This chapter presents, in a practical way, the most well–known quantum algorithms: Deutsch's algorithm, Simon's algorithm, Shor's algorithm, and Grover's algorithm. These algorithms are the most important component of the theory of quantum computing –in our view– and deserve a clear, accurate treatment. The account given in this book is readable but brief; the interested reader will have to resort to more specialized texts [6, 2] for explanations and details. The dense coding scheme, and the BB84 protocol for quantum key distribution, are also covered in this chapter. This is misleading as neither of these is regarded as a quantum *algorithm* in the accepted sense of the term.

Chapters 22–23: Quantum Information Theory and Quantum Error Correction Chapter 22 presents a selection of results from quantum information theory, such as measures of entanglement and the Holevo bound. The subsequent chapter deals with the 5–, 7–, and 9–qubit error correction codes. The treatment of all these topics is concise.

Chapter 24: Quantum Hardware This final chapter details proposed physical implementations of quantum computational devices, including cavity QED and NMR spectroscopy.

A list of websites related to quantum computation and quantum information is provided in the form of Chapter 25, and this is followed by bibliography and index.

3 Strengths

This book is essentially a compendium of techniques and computer programs usually dealt with in undergraduate computing courses. It is handy as a reference, since it covers a wide range of topics concisely. Its coverage of genetic algorithms, and gene expression programming in particular, is novel and useful for beginning researchers. The material on quantum computing has never appeared in a general computer science text before, and serves as a convenient summary of the basics; it is quite brief compared to other texts. The C++ and Java code provided is useful in practice, and helps to illustrate the techniques discussed.

4 Weaknesses

This textbook has two particular weaknesses, in our view. On one hand, the book covers far too many topics, and does little justice to most. On the other hand, while the title may suggest a balanced treatment of classical and quantum computation theory, the book devotes much less space to quantum computing and quantum information.

This book scores very highly on breadth of coverage, but at the cost of depth. Furthermore, the amount of space devoted to each topic is disproportionate. Topics of especial importance in computer science, such as computability and complexity, boolean algebra and cryptography (discussed in 8 pages, 23 pages and 6 pages respectively) receive far less attention than neural networks (35 pp.) and genetic algorithms (89 pp.). Clearly, an element of balance is missing.

The bibliography contains significantly more items related to quantum computation and quantum information than items related to classical computer science. This mismatches the proportion of material in the book on classical and quantum computing.

To address the question of the book's intended audience, we resort to the book's preface, where we are told that:

> "Scientific computing is not numerical analysis, the analysis of algorithms, high performance computing or computer graphics. It consists instead of the application of these fields and others to craft solution strategies for applied problems. It is the original application area of computers and remains the most important. [...] More and more universities introduce a Department of Scientific Computing or a Department of Computational Science. [...] This book can serve as a text book in Scientific Computing. It contains all the techniques (including quantum computing)."

The above fragment contains a couple of fallacies, if we are to accept the conventional usage of the term *scientific computing.* Textbooks and university curricula for the subject known as scientific computing, almost without exception, contain material traditionally associated with numerical analysis. What is more, the last sentence in the fragment seems to suggest that quantum computing is a 'technique' which belongs to this subject; again, this depends on your definition of 'scientific computing.'

So let's make this clear: *scientific computing* is a term that normally refers to the study of such topics as linear and nonlinear equations, least eigenvalues, optimization, integration, interpolation, ordinary and partial differential equations, and fast Fourier transforms. This is not the subject of Hardy and Steeb's text at all. To drive this point home, I will quote from [4], a highly regarded text on scientific computing:

> "The subject of this book is traditionally called numerical analysis. Numerical analysis is concerned with the design and analysis of algorithms for solving mathematical problems that arise in many fields, especially science and engineering. For this reason, numerical analysis has more recently also become known as scientific computing."

The authors seem to suggest that a 'Department of Scientific Computing' and a 'Department of Computational Science' are one and the same. Computer Science departments generally do not confine themselves to the study of numerical analysis, no matter what they are called. The distinction has been made particularly clear here at the University of Warwick, which actually has both a 'Department of Computer Science'[3] *and* a 'Centre for Scientific Computing'[4].

[3]See `http://www.dcs.warwick.ac.uk/`

[4]See `http://www2.warwick.ac.uk/fac/sci/csc/`

Putting the semantics of the term 'scientific computing' aside, we return to the question of this book's intended audience. This book is very usable as a reference on a number of topics, and someone with a mild interest in quantum computing may consider it a handy addition to their bookshelf. Apart from quantum computing, the book has good chapters on neural networks and genetic algorithms, but researchers in any one of these fields will only use it to recap on the basics, while referring to more specialized texts for details. Undergraduates may find some use for the C++ and Java code, but the book does not help as a text for a specific course. That is why, in our view, the readership for this book is difficult to determine.

Finally, we should point out that C++ and Java code listings are often several pages long; we feel that printing code fully in the book is not necessary, since anyone who is interested in the code can download it off the web and peruse it on his/her own machine.

5 Conclusions

Our criticisms of this book do not imply that its authors have not made a considerable effort; indeed, to cover such a wide range of topics, especially in a short amount of space, requires a lot of skill. We express the hope that a future edition of the book will address some of the issues raised here, and reach a wider audience; also, we welcome the initiative to present the subject of quantum computation and quantum information in a non–specialized text. Hardy and Steeb have already produced a brilliant little book [8] with solved problems in quantum computing and quantum information, which we highly recommend.

References

[1] Gay, S., Nagarajan, R., and Papanikolaou, N. Probabilistic model–checking of quantum protocols. Quantum Physics Repository Preprint quant-ph/0504007, available at www.arxiv.org.

[2] Gruska, J. *Quantum Computing.* McGraw–Hill International, 1999.

[3] Hardy, Y., and Steeb, W.-H. *Classical and Quantum Computing With C++ and Java Simulations.* Birkhäuser Verlag, 2001.

[4] Heath, M. *Scientific Computing*, 2nd ed. McGraw–Hill, 2001.

[5] Nagarajan, R., Papanikolaou, N., Bowen, G., and Gay, S. An automated analysis of the security of quantum key distribution. CoRR Preprint cs.CR/0502048, available at www.arxiv.org.

[6] Nielsen, M. A., and Chuang, I. L. *Quantum Computation and Quantum Information.* Cambridge University Press, 2000.

[7] Papanikolaou, N. Introduction to quantum cryptography. *ACM Crossroads Magazine 11.3* (Spring 2005 Issue).

[8] Steeb, W.-H., and Hardy, Y. *Problems & Solutions in Quantum Computing & Quantum Information.* World Scientific, 2004.

Review of
Integer Programming [5]
Author of book: Laurence A. Wolsey
Publisher: Wiley and Sons 1998
$94.95, 265 Pages
Author of Review : E. Jonathan Chapin

6 Introduction

Laurence A. Wolsey's book Integer Programming is one of the few books currently available which is dedicated to the subject of integer programming. As such, it is a very well written and comprehensive book on the subject, and is written in a manner which is friendly to those who are new to the subject. It is reminiscent of a good advanced mathematics textbook, as it comprehensively explains and proves its topics, as well as offering some examples and problems for the reader to solve. However, perhaps more due to the complexity of the subject matter itself rather than the ability of Wolsey to convey said subject matter, it is not an easy read for anyone interested in learning about Integer Programming. An appropriate audience would be graduate computer science students with a strong background in math and mathematical proofs using it as a course text, or postgraduates looking to learn the subject matter on their own. There are several prerequisites to reading the book that one should have in order to come away with a good understanding. The most basic need is for an understanding of mathematical notation and proofs, as they are (unsurprisingly) used throughout and often as explanations of themselves. One should also have a good grasp of linear algebra and topology of the real numbers, as they are both commonly used in the material and integral to the concepts explained. It is suggested that the reader have at least a basic understanding of linear programming as well, as several of the beginning and almost all of the later techniques and problems make use of it, and Wolsey does not explain how to solve such and uses some limited terminology related to the subject. Integer Programming is an in depth book. It is not a book intended to be a quick guide to integer programming techniques and applications and should not be taken as such. This book is intended to give a comprehensive understanding of the problems of integer programming and its related theories. The reader should come away from this book with an understanding of how to tackle problems related to integer programming, and also understand the complexity of the problems as well as the complexity, efficiency, and theories behind the techniques used to solve such problems. The material is comprehensive and not easy to take in though, and may require several re-readings in order to wholly understand some of the concepts presented.

7 Summary of Chapters

The first chapter is devoted entirely to explaining exactly what an integer program is, and how to formulate one. A standard (mixed) integer program, or MIP, is to maximize (or minimize) an equation over several variables, given a set of general constraints related to those variables and the constraint that at least one of the variables must be an integer. Mathematically speaking: Given variable vectors x and y, constant vectors c, h, and b, constant matrices A and G max $cx + hy$ subject to: $Ax + Gy \leq b$. $x \geq 0$ $y \geq 0$ and integer.

[5]©2005 E. Jonathan Chapin

The chapter explains some further subsets of integer programming, as well as the fact that many Combinatorial Optimization Problems can be rewritten as integer programs. This chapter also gives several examples of problems that can be solved using integer programs, and shows how to formulate these problems into integer programs. It also covers some basic facts about the complexity of these problems, and introduces the idea that initial *direct* formulations often can be changed or replaced by alternative formulations of the same problem, and that these alternate formulations can be *ideal* (covering the smallest subset of a solution-space while still containing all of the valid solutions). Overall this chapter offers a nice and simple introduction to the material. The second chapter covers the fact that we simply can not get a solution and claim it to be optimal, but rather that it should be provably so. It also covers the technique of problem relaxation, in which we can take an initial hard problem, and by removing constraints or changing some, form a much easier problem to solve. Doing so brings us to the final idea of the chapter, that we can get a better idea of what values we're looking for through either finding a merely feasible solution, or by using a simpler problem to provide us with a bound for the solutions of the original problem. This chapter is a little more technical and fast paced than the first, but is overall still a relatively simple chapter from the standpoint of coverage and readability. The third chapter studies some of easier and more contrived problems related to integer programs. All of these problems have an alternative known method for solving the problem which is *efficient*. While this may not seem to be as worthwhile or interesting as other problems the reader may be interested in solving, it follows the book's apparent course of absolutely covering the material in every sense. This chapter provides good examples to study for students new to some of the concepts, and a fresh perspective on these problems to the more experienced readers. The fourth and fifth chapters is where the book switches to more advanced material as well as going at a faster pace. The fourth chapter covers concepts such as taking large disjoint edge sets and using augmented paths created from them to make larger disjoint sets. The fifth quickly covers dynamic programming (this is *not* the same Dynamic Progrmming that is a pardigm in algorithms) as a bottom up heuristic method, and its use as an alternative way to find solutions to some problems. These chapters cover the material both in depth and much more from a theoretical point of view as well as giving examples in a more applied mathematics sense. It also doesn't go over as many examples as previous chapters and tends towards a minimum explanation while still managing to be comprehensive. Students who had a few problems with previous chapters may start to struggle here. All the material needed is there, but it is easy to get *behind* if the reader continues while not entirely understanding an earlier concept. Chapter six is a bit of a departure from the other chapters in that it is, for the most part, purely theoretical and abstract. It covers the concept that there are problems that can be solved in polynomial time (P problems like shortest path) and NP problems: problems for which no efficient polynomial algorithm has been found, in which general integer programming problems fall. The point of this chapter seems to be to illustrate that the only known sure approach to solving many of the problems we are concerned with is a brute force test of all the feasible solutions, for which there usually is an exponentially large set. This leads to the newer, less orthodox, and not guaranteed to work techniques of later chapters. This chapter could be skipped as far as understanding the techniques themselves, but should be included for those seeking a true understanding of the *why* of integer programming techniques. Chapter seven marks the start of teaching somewhat general methods of solving general integer programs. The technique described here is known as *branch and bound*, which is a way of using linear relaxation of an integer program and then branching roundings of the results to come up with a direct way to find an optimal integer solution. While this can be an effective method, it actually may never find a feasible integer solution (this is NP hard), so some modifications such as adding heuristics or optimizing the linear relaxations as they are made are discussed in light detail. This chapter is more of what someone looking just for *how* would want, as it explains the algorithm in good detail and gives a few short of examples of it being used. This is a step down from previous chapters in the sense of having to comprehend mathematical complexities from brief though thorough explanations, and for students may turn the tables of who grasps the concepts right away and who may struggle with the material a bit. The eighth and ninth chapters cover the use of cutting planes, which is a technique in which one can remove redundant constraints in favor of dominant constraints, as well as adding new constraints based on the integer nature of IPs in order to further reduce the search field. These chapters go back to the earlier balance of material, where mathematical proofs of the concepts are given, as well as explanations and examples as to how to use the techniques described. These chapters strike the best balance in the book as to both covering the concepts behind the material as well as the material itself. The tenth, eleventh, and twelfth chapters cover much more complex techniques for approaching integer programming problems. Chapter ten covers LaGrangian relaxation, in which we separate the *complicating* constraints from the *easy* ones, and reformulate the objective function using the complicated constraints as reformed variables in that part of the equation. Chapter eleven takes on the grandiose task of solving the most general form of an integer program using the linear relaxation of this master integer program and column generation algorithms. Finally, chapter twelve covers many complex heuristic algorithms for varied types of problems in a near rapid-fire succession. Overall, these chapters are where some of the most useful material comes from, as well as some of the most difficult. These chapters should be taken slowly and read several times in order to take in the vast amount of complex ideas and techniques covered. In the final chapter, chapter thirteen, we finally go into the practice of integer programming. Appropriately enough, this chapter is titled *From Theory to Solutions.* After a very brief mention of the use of modeling languages for problems, Wolsey brings most of the material from previous chapters together to solve (or at least come up with good algorithms for solving) some of the *hard* problems mentioned in previous chapters. This part is the most application-related part of the book, as many of the problems are related to real business and technology situations, such as manufacturing and networking. This chapter finally brings the kind of instruction the more application-minded may have wanted, while at the same time illustrating just how necessary all the theory side information is to the solving of these problems.

8 Opinion

Overall, Wolsey's Integer Programming sets the bar for what future integer programming books should be, and sets it high. It would be a great choice for a student text to teach the subject from, and a good choice for learning the material as comprehensively as possible on one's own. While it is certainly not a simple text, the level of the material matches the level of complexity in which it is explained. The only thing found wanting in this text is for some more of some of the concepts to be explained a bit less quickly and more clearly, but otherwise, it comes highly recommended for anyone wishing to truly learn about integer programming.

Computational Line Geometry
Authors of Book: H. Pottmann and J. Wallner
Publisher: Springer Verlag, 2001, 575 pages, $98.00, Hardcover
Reviewed by Georg Essl [6]

9 Introduction

How does the configuration of hinges of the limbs of a robot impact its motion? How can I describe visible phenomena from optics and mechanics? Will my packaging machine jam or create unattractive creases? How can I reconstruct objects that are mathematically simple to manipulate from scanned data?

All these are rather applied questions and all these questions have to do with geometry. Some of these problems are new. They came about because of advances in mechanical engineering and data acquisition. Others are very old problems, like the attempts to explain focal lines created by lenses.

Computers have in many ways altered how we can ask questions and find answers about such problems. While in the past "tracing curves" was a skill to acquire, this has become a task relegated to computer graphics. Instead we may want to worry about computational structures and algorithmic complexities of the algorithms to do this.

We do have a vast literature on numerical methods of geometric objects based on splicing our objects into discrete meshes and treating discrete mesh points. This immediately gives the data structures and it allows us to ask questions of efficiency and accuracy. This body of knowledge typically deals with objects in Euclidean space with Cartesian coordinated and discrete approximations of differential operators on these objects. Literature on visualization and numerical methods in computer graphics usually follow this approach.

Another approach is concerned with the computation of discrete and combinatorial structures in relationship to some geometrical situation. These structures again lend themselves to complexity analysis. How can a space best be partitioned? How can we find points of collisions efficiently? This is what many of us would understand to be "Computational Geometry".

As with most classifications there is some overlap.

9.1 Classical Geometry and Line Geometry

The 19th century was a period of rapid development of new concepts in mathematics and geometry. This literature is so rich that it is not clear if even today all these ideas have been properly assimilated into our working literature and college teachings.

This period brought the advent of algebraic/analytic geometry of Poncelet, Steiner, Plücker, Clebsch, Klein, Lie, Cayley and others, of differential geometry of Riemann and topology à la Poincaré. Today we think of algebraic geometry as algebraic varieties of sheaves and schemes of Grothendieck. Differential geometry is the study of differential forms on manifolds after Cartan. What contemporary Riemannian geometry and concurrent developments in topology have in common, is that "classical" knowledge is well present in the modern formulation. But what about the algebraic/analytic geometry of present and past? Here the modern revolution of abstract algebraisation has separated these two by much. In many ways the classical results are too concrete to fit well with the abstraction. Secondly, the emphasis has drastically shifted towards the algebraic content of the body of knowledge and away from geometric arguments.

[6]©Georg Essl, 2005

To be accurate, I cannot call a good part of the 19th century developments algebraic or analytic. Many ideas came originally about in attempts to pursue geometry synthetically. The driving approach was an attempt at getting to theorems by synthetic geometric construction. But since the discovery of the intimate relationship between linear algebra and projective geometry much of this work is since been moved into an analytic context.

Plücker promoted the idea of studying geometry in three dimensions using lines as basic elements [6]. It was a major motivator of works to come by Lie, Klein and others. While lines as primary objects of the study of geometry were around before, Plücker has made it into a field of study. A flurry of texts came out until 1900, but in many ways development has stalled since the 1920s, when Zindler wrote a comprehensive survey for the Encyklopädie der Mathematischen Wissenschaften [7]. Ever since the publications of texts on the topic has been rather sparse.

What is this body of work and how can it help solve our rather applied problems?

10 The Book

The book under review carries the title "Computational Line Geometry" written by Helmut Pottmann and Johannes Wallner and published in 2001 in the series Mathematics and Visualization by Springer Verlag.

10.1 Scope

In the introduction, the authors promise to introduce *line geometry* in the sense of Plücker and beyond with a target audience of graduate students and researchers in science and engineering with an interest in geometry and geometric computing and its applications. They explicitly state that neither data structures nor algorithmic complexity are treated and refer to an article by Chazelle and co-workers for details. This turns out to be correct and one will be hard pressed to find traditional computational geometry results in this text. This is rather important. When I first read the title of the book I had two possible interpretations of it in mind. One was "Computational Geometry of Lines" and the other was "Numerical Line Geometry". Given the authors' stated intent and the content of the book, I was mostly wrong in both cases. The book is written with a good eye towards applications. Hence computations are presented in an explicit form and can usually be readily implemented. I assume this is what the authors mean with "computational".

10.2 Content

The book consists of 8 chapters spread over 546 pages, 222 references at the end, a list of symbols and a 7-page index. There are no exercises provided.

10.2.1 Chapter 1: Fundamentals

The authors give a clear, readable and reasonably detailed account of projective geometry with emphasis on the real case. The notation is clear though slightly different from other expositions (for example Coxeter's [3]), but there hardly seems to be a standard notation in the literature. Then there is a brief exposition of projective differential geometry. It's less than a third of the length of the section of projective geometry. Differentiable results and methods are a minority in this text which may justify this brevity. For details it should at least be supplemented by a text on differentiable curves, for example Bruce and Giblin [2]. Then a very quick introduction to algebraic geometry with an eye towards computability. The authors introduce just enough language as needed later and swiftly get to Bézout's Theorem of the intersection numbers of algebraic curves. The treatment is completely pre-pre-sheaf-theoretic, if the pun is allowed. A similar scope at a much more leisurely pace (minus the introduction of the Gröbner basis) can be found in the elementary text by Gibson [5], which may help supplement the basics and Fischer's recent text on plane algebraic curves [4]. The chapter wraps up with a clear treatment of Bézier curves and surfaces.

10.2.2 Chapter 2: Models of Line Space

The authors introduce coordinate and their corresponding manifold representations for lines in projective space. Refreshingly, the author's treatment emphasizes coordinates and geometry over algebraic development. This chapter is foundational for much what is to follow, except for a brief introduction to linear forms and Grassmann Algebra, which is presented mostly for reference.

10.2.3 Chapter 3: Linear Complexes

The properties of the set of lines by a linear homogeneous equation in Plücker coordinates called linear complexes are discussed in detail. In particular these turn out to relate be helical motions in Euclidean space. These properties are then illustrated with respect to kinematics. Motion in Euclidean space can be described via the trajectory normals, which in turn are linear complexes projectively.

10.2.4 Chapter 4: Approximation in Line Space

How can one use linear complexes to approximate a pre-defined configuration? Using quadratic norm minimizers, a range of solutions are offered here. Particularly nice is the treatment of shape reconstruction. The coverage of stability questions of linked rigid bodies, very relevant for robotics, is probably too short to be useful without additional literature, in particular with respect to singularity theory.

10.2.5 Chapter 5: Ruled Surfaces

Ruled surfaces are the center piece of projective differentiable geometry. They are line families defined as curves in Plücker coordinates. Depending on the nature of these curves one deals with a differentiable or an algebraic case. The authors treat both cases. Then these results are related back to Euclidean geometry which nicely illustrates the practicality of ruled surfaces. To numerically implement dense or differentiable line sets discuss the discrete case, the authors provide one of the rare spots in the book which connects to traditional numerical and computational geometry. In this context the authors breeze through a number of examples including variational design and the important case of offsets (outside the computer aided graphic design community these are maybe more familiar as "parallels").

10.2.6 Chapter 6: Developable Surfaces

Developable surfaces are those than can be deformed from the Euclidean plane. This could be studied by singularity theory of Whitney maps. The authors offer here a related though rather applied formulation of this. They also bring results of developable surfaces back to classical results of differentiable curves leading to example applications to geometrical optics. In this context they use the less common word anticaustic instead of orthotomic, but the treatment should be clear for the reader familiar with the latter. This goes along with a treatment of Phythagorean hodograph curves as widely applied recently by Farouki and co-workers.

10.2.7 Chapter 7: Line Congruences and Line Complexes

Here the authors return to congruences and complexes this time of lines in algebraic and differentiable setting. The notion of congruence is indeed important to geometrical optics and the authors develop applications of the results to this end. A short section discusses briefly numerical aspects.

10.2.8 Chapter 8: Linear Line Mappings — Computational Kinematics

The behavior of linear mappings in Plücker coordinates are discussed with interesting applications to packing and planar surveying. The action of quaternions is introduced to discuss spherical motions.

10.3 Presentation

Wolf Barth wrote in an introduction to lecture notes on the geometry of circles that a text on geometry should have a picture a page [1]. The text under review does not quite reach this measure, yet, in my mind, achieves its spirit. The text contains 264 figures over 546 pages. They are very carefully crafted and illustrate the situation very clearly. I did not come across a figure that I feel was unclear or misleading. This effort is to be applauded.

Some figure captions could be improved. For example some captions of figures with multiples sub-figures describe the right sub-figure before the left, against the typical direction of reading and hence potentially confusing the reader.

The organization of the book works reasonably well. The index is comparable to other graduate texts, which in my personal evaluation is just about acceptable, though I wish the standard overall would be somewhat higher. A number of basic terms used in the text are missing, yet the index is reasonably complete and leads to sensible locations in the text.

11 Conclusions

Overall I recommend this text to anyone who wants to learn about line geometry, projective geometry and the geometric side of some of algebra. The book fills a niche that has been neglected for long and should benefit researchers interested in geometric methods. This text will however not satisfy the reader seeking an undergraduate introduction. The book's content is too advanced for undergraduates and not structured to be easily usable as a textbook, as exercises are missing. Nor does it use much of the contemporary abstract language and algebraic treatment, hence may not satisfy the more algebraically minded reader. There is also little emphasis on computational geometry. The text also doesn't include much in terms of explaining historical developments of ideas. Here I would refer the reader to Zindler's Encyklopädie survey [7], regrettably only available in German. The authors also do not provide references to more abstract treatments and related concepts (for example relationships to Lie algebras, or contact geometry are not pointed out). None of these are stated goals of the authors and the book should not be faulted for their absence.

This book offers a detailed look at line geometry with emphasis on applicability, which should be of great value for graduate students and researchers, in particular in Computer-aided Geometric Design. It should also give new opportunities and perspectives for developments in Computer

Graphics, Computational Geometry, Kinematics, Robotics and geometrical formulations of problems in the natural sciences, for example Optics and Mechanics. It covers a body of knowledge that is underrepresented in the literature and deserves to be known more widely.

The authors wrote a clearly developed and beautifully illustrated book that fills a gaping hole in the contemporary literature.

References

[1] W. Barth. Kreise. Lecture Notes, `http://www.mi.uni-erlangen.de/barth/docs/kreiset1.ps`, 1997.

[2] J. W. Bruce and P. J. Giblin. *Curves and Singularities.* Cambridge University Press, Cambridge, UK, 1992.

[3] H. S. M. Coxeter. *Projective Geometry.* Springer, New York, second edition, 1987.

[4] G. Fischer. *Plane Algebraic Curves.* American Mathematical Society, Providence, Rhode Island, 2001.

[5] C. G. Gibson. *Elementary Geometry of Algebraic Curves.* Cambridge University Press, Cambridge, UK, 1998.

[6] J. Plücker. *Neue Geometrie des Raumes gegründet auf die Betrachtung der geraden Linie als Raumelement*, volume 1,2. Teubner, Leipzig, 1868.

[7] K. Zindler. Algebraische Liniengeometrie. In *Encyklopädie der mathematischen Wissenschaften mit Einschluss ihrer Anwendungen.* Teubner, Leipzig, 1921.

Logic for Computer Scientists
by Uwe Schöning
$50.00, 175 pages, Hardcover
Birkhauser, 1994

Review by Riccardo Pucella[7]

This is a short introductory book on the topic of propositional and first-order logic, with a bias towards computer scientists. What's a bias towards computer scientists, you ask? Good question. Despite my initial belief, it does not mean that it introduces logic with an eye towards the wide variety of areas where logic is used in computer science. Rather, it gives a computational perspective on logic; the emphasis is on computational aspects of logic, and specifically on procedures for establishing the satisfiability or unsatisfiability of formulas, culminating in various forms of the resolution procedure. (In that same sense, Smullyan [5] gives a tableaux-based perspective on first-order logic.) The study of resolution procedures leads naturally to the topic of using predicate logic as a programming language, that is, logic programming.

This focus on computational aspects means that there is much less coverage of those topics typically found in logic textbooks, such as axiomatization and completeness results, or model theoretic notions such as applications of compactness and theories. This is not a criticism, mind you: there are other good introductory books that deal with that. Schöning decides to concentrate on computational issues, and gives us a short book (less than 170 pages) with a tight storyline.

[7]©Riccardo Pucella 2005

Summary of content. The book consists of three chapters.

Chapter 1 is about propositional logic. After the obligatory introduction to the syntax and semantics of propositional logic, the two standard normal forms for formulas are presented, namely disjunctive and conjunctive normal forms. The special class of Horn formulas is introduced, and an efficient algorithm for deciding satisfiability of Horn formulas is given. The compactness of propositional logic is then established: a set of formulas is satisfiable if and only if every finite subset is satisfiable. Building on this result, the resolution calculus for propositional is defined: a set of rules that can be applied to clause sets (an alternate representation for a formula in conjunctive normal form) to derive new clauses. The resolution calculus is proved to be refutation complete: if the empty clause is derivable in the calculus, then the original clause set is unsatisfiable, and if the original clause set is unsatisfiable, then the empty clause is derivable. This gives a procedure for deciding satisfiability of propositional formulas. (A specialization of the resolution calculus to Horn formulas, unit resolution, appears in Exercise 35.)

Chapter 2 is about first-order logic. After the presentation the syntax and semantics of first-order logic, normal forms for first-order formulas are introduced: prenex form (where all quantifiers are in the front of the formula), Skolem form (where all existential quantifier have been removed and all existentially quantified variables replaced by a function of the remaining variables), and a clause set representation extending that of propositional logic. The undecidability of the validity problem for first-order logic is proved by reduction from the Post correspondence problem. (The decidability of monadic predicate logic appears in Exercise 69.) A quick mention of theories is made, but not pursued. The important class of term models (also known as Herbrand structures) is introduced, and used (in passing) to prove the Löwenheim-Skolem theorem. The Herbrand expansion of a first-order formula, which produces an infinite propositional clause set by replacing every variable of the formula in its clause set representation by an element of the term model, is defined, and compactness of propositional logic is used to prove Herbrand's theorem: a first-order formula is unsatisfiable if and only if there exists a finite subset of its Herbrand expansion that is unsatisfiable. As an application of Herbrand's theorem, a procedure due to Gilmore that enumerates the clauses of the Herbrand expansion of a formula and thus provides a semi-decidable procedure for the satisfiability problem of first-order logic is presented. Gilmore's procedure is then modified to use propositional resolution as its main step, yielding the ground resolution procedure for first-order logic. After analyzing the difficulties with this procedure, the resolution procedure of Robinson is described; it works directly on first-order clause sets, and requires unification of first-order terms. This resolution procedure is shown to be complete for first-order formulas. A numbers of refinements of the general procedure are presented, along with completeness results; these include unit resolution, which is defined for all formulas, but is complete for (first-order) Horn formulas, and SLD resolution, which is defined only for Horn formulas, and is also complete for them.

Chapter 3 is about logic programming. More precisely, it studies the problem of turning first-order logic into a programming language. The idea is well known. Start with a *query* formula of the form $\exists x.P(t,x)$, where t is a variable-free term. Clearly, this formula is valid if and only if $\forall x.\neg P(t,x)$ is unsatisfiable. A resolution refutation of $\forall x.\neg P(t,x)$ showing it unsatisfiable can be modified to yield a c such that $P(t,c)$ holds; this c is called an *answer* to the query. In this way, it is possible to view first-order logic as a query language driven by the resolution procedure. For efficiency reasons, this language is often restricted to Horn formulas, with appropriate restrictions to the resolution procedure (such as using SLD resolution), and using further special evaluation strategies to resolve the nondeterminism inherent in the resolution procedure. The programming language Prolog is described; it is probably the best known practical language that uses Horn formulas with a restricted resolution procedure.

Opinion. I found this a nicely written book with many examples and exercises (126 of them). The presentation is natural and easy to follow. It should be noted, however, that due to both its length, and its date of publication (the first printing of the English edition was in 1989), recent developments in logic programming are not included. Since this book is meant to be introductory, it is fair to ask how it relates to other "popular" introductory books on logic. Well, what do I have on my bookshelf that compares? Let's leave aside the two books from which I learned logic, Enderton [1] and Shoenfield [4], which are both much more mathematical. A book that is much closer in spirit is Nerode and Shore [3], who also in some sense target computer scientists. There is an overlap in topics, as Nerode and Shore also discuss resolution, and much more. Schöning's book holds up surprisingly well to Nerode and Shore's, in no small part due to it being shorter and much more focussed. Finally, going back to my initial belief prompted by the title of the book, a good textbook that describes the uses of logic in computer science, with an emphasis on verification, is that of Huth and Ryan [2].

This book seems suitable for a short course, a seminar series, or part of a larger course on Prolog and logic programming, probably at the advanced undergraduate level. For a general logic course, even in computer science, it should be supplemented to give a more complete picture of the subject.

References

[1] H. B. Enderton. *A Mathematical Introduction to Logic.* Academic Press, 1972.

[2] M. Huth and M. Ryan. *Logic in Computer Science: Modelling and Reasoning about Systems.* Cambridge University Press, 1999.

[3] A. Nerode and R. Shore. *Logic for Applications.* Springer-Verlag, 1994.

[4] J. R. Shoenfield. *Mathematical Logic.* Addison Wesley, 1967.

[5] R. Smullyan. *First-Order Logic.* Springer-Verlag, 1968.

Review of
Teaching Statistics Using Baseball [8]
Author: Jim Albert
Publisher: MAA, Year:2003, Price: $46.95, Softcover

Reviewer: James Glenn

12 Introduction

Looking for a familiar context in which to teach statistics, Jim Albert turns to baseball to take advantage of its plentiful statistics and rich history in his book *Teaching Statistics Using Baseball.* His book covers analyzing a single batch of statistics, comparing batches of statistics, and analyzing relationships between statistics. While it may fall short of being a home run, it is at least a double

[8] © James Glenn, 2005

off the top of the wall; it provides a wealth of examples and exercises to aid baseball fans who teach statistics and would make an excellent supplemental textbook.

13 Summary of the Book

The first chapter is an introduction to baseball statistics using Rickey Henderson's career stats as an example; this example is used in at least one exercise in each chapter throughout the rest of the book. It is not really a true introduction of baseball statistics; rather it is an introduction to statistics using baseball as an example. Students unfamiliar with the terms "at bat", "base hit", or "on base percentage" will have to turn to an appendix for an explanation. However, even the appendix may not be enough to prepare students unfamiliar with baseball for the onslaught of baseball jargon that will follow. A second appendix gives links to websites that provide baseball statistics; the book has a 2003 copyright and as of 2004 all of the sites are still running.

Each of the remaining chapters is laid out in the same way: several well-presented and well-motivated case studies of generally increasing complexity are given. Each case study begins with a list of the topics covered, although these topics are not always explicitly mentioned in the case study; presumably they would be explained in the primary textbook. The end of each chapter provides numerous appropriately chosen exercises.

Chapters 2 and 3 introduce qualitative and simple quantitative analysis of a single batch and then multiple batches of statistics using histograms, stem plots, five-number summaries and box-plots, time series plots, and dotplots. It becomes clear from these first chapters that the book is unsuitable for use as a primary source in a rigorous statistics course: "mean" is used but never defined; the definition of "median" is first given parenthetically. Likewise, there is an excellent discussion of standardization in the context of comparing extreme statistics from different seasons (that is, is George Brett's .390 batting average in 1980 as impressive as Rod Carew's .388 in 1977?), however any rigorous treatment of the normal distribution will have to come from another source. We also find here evidence of the richness of baseball as the context for a statistics course: after reading the brief discussion of run support I thought of a follow-up that would integrate very well with a later chapter.

Chapter 4 explores relationships between measurement variables through linear regression, including multiple linear regression. The motivating question for much of this chapter is "what moves can a team make in the off season to increase the number of runs it scores?", or, in other words, "what offensive statistics correlate best with runs scored?" One case study shows that team on base percentage is a much better indicator of runs scored than team batting average. The next step is to use multiple linear regression to determine what linear combination of offensive statistics is the best indicator of runs scored. Any baseball fan with an interest in statistics will find this fascinating, although the author's result uses data from just one season and so is not quite "reasonable" – I wish that in the end we got the "reasonable" results instead of just a pointer to them. The emphasis is again on how to use statistics rather than on a rigorous discussion of the theory behind them.

Chapter 5 is an introduction to probability using tabletop baseball simulation games. This is perhaps the least interesting chapter in the book, but it will suffice. Chapter 6 follows with a discussion of the binomial distribution (with formulas for once!) and an interesting model of scoring runs using an "on base profile."

Chapters 7 (inference) and 8 (topics) both rely heavily on the use of simulations. In chapter 7 simulations are ultimately used to find probability intervals for players' true abilities (rather than their observed results). Chapter 8 uses simulations to explore situational effects (for example, batting average in home versus away games and before the All-Start break versus after). Here the author explains how to construct simulations to test whether such effects are the result of chance alone, bias, or ability.

Chapter 9 is perhaps the most intriguing of all. Here, the author uses Markov chains to determine the "run potential" of a state, which for this example is a combination of the number of outs and the runners on base. For instance, having a runner on first and no one out is worth, on average, 0.85 runs for the remainder of the inning. These results are used to analyze various baseball strategies. A logical follow-up, not presented in the text, is to use these results to evaluate the offensive performance of hitters.

14 Opinion

None of these chapters will stand on their own as a source for a rigorous statistics course. Those who have never seen a game of baseball will need help keeping up with all of the jargon. Still, as a supplement the book is excellent. Statistics instructors will find much of the course preparation done: the case studies provide the starting point for many hours of lectures and the exercises should suffice for any homework assignments. The enthusiastic baseball fan who teaches statistics undoubtedly will be inspired to expand the material in new directions for additional lectures or projects. Those fans who are taking a statistics course may find this book useful for reinforcement or project ideas regardless of whether their course is centered on baseball. Finally, mathematically inclined baseball fans will find many of the case studies fascinating even if they are not taking or teaching a statistics course.