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. **Quantum Computation and Quantum Communication: Theory and Experiments**. Author of book: Mladen Pavicic. Author of review: George Hacken. This is a book on quantum computing with more physics in it then usual. To quote the review: "Pavicic's book achieves its own characteristic balance between the complementary attributes, depth and breadth, such that algorithmists and computation theorists will be reading both within and very much around their subject as that subject is conventionally construed."

2. **Quantum Computing for Computer Scientists** Authors of book: Noson S. Yanofsky and Mirco A. Mannucci. Author of review: S.C. Coutinho. I parapharse the review: This is an introduction to quantum computing and quantum cryptography where the required background in physics is kept to a minimum. The introduction is limited to finite dimensional quantum systems. This also means that a solid knowledge of linear algebra will be enough for reading the book.

3. **Biologically Inspired Algorithms for Financial Modelling** . Authors of book: Anthony Brabazon, Michael O'Neil. Author of review: Brad G. Kyer. To summarize this book I quote the review itself: "If one takes the approach that the market behaves much like a living, breathing, biological ecosystem then analogously biologically inspired algorithms may prove to be more effective tools. The objective of this book is to provide an introduction to biologically inspired algorithms and some tightly scoped practical examples in finance."

4. **The Handbook of Bioinspired Algorithms and Applications** Editors of Book: Stephan Olariu and Albert Y. Zomaya. Author of review: Kushal Chakrabarti. This book provides a starting point for researchers wanting to apply biologically-motivated algorithms to their specific problems.

5. **Theoretical and Experimental DNA Computation.** Author of book: by M. Amos. Author of review: Maulik A. Dave. I quote the review: "The book is an overview of DNA computing. It touches both theoretical and experimental aspects. The theoretical aspects include algorithms, and computing models. The experimental aspects consist of various DNA experiments done for computing."

6. **Coding for Data and Computer Communications** Author of book: by David Salomon. Author of review: Fatima Talib. This book looks at reliability, efficiency, and security in data storage and transmission. The book lies in the intersection of information theory, cryptography, networking, and information and communication technology.

[1]© William Gasarch, 2009.

The following books are available for review. If you email me a request to review it and your postal address I will email you guidelines for reviewers and postal mail you the book.

Books on Algorithms and Data Structures

1. *The Art of Computer Prgramming Vol 4, Fascicle 0: An introduction to Combinatorial Algorihtms and Boolean Functions* by Donald Knuth
2. *Algorithmic Adventures: From Knowledge to Magic* by Juraj Hromkovic.
3. *Matching Theory* by Lovasz and Plummer.
4. *Algorithms and Data Structures: The Basic Toolbox* by Mehlhorn and Sanders.
5. *The Algorithms Design Manual* by Skiena.
6. *Algorithms on Strings* by Crochemore, Hancart, and Lecroq.
7. *Combinatorial Geometry and its Algorithmic Applications: The Alcala Lectures* by Pach and Sharir.
8. *Algorithms for Statistical Signal Processing* by Proakis, Rader, Ling, Nikias, Moonen, Proudler.
9. *Nonlinear Integer Programming* by Li and Sun.
10. *Binary Quadratic Forms: An Algorithmic Approach* by Buchmann and Vollmer.
11. *Time Dependent Scheduling* by Gawiejnowicz.
12. *The Burrows-Wheeler Transform: Data Compression, Suffix Arrays, and Pattern Matching* by Adjeroh, Bell, Mukherjee.
13. *Parallel Algorithms* by Casanova, Legrand, and Robert.
14. *Mathematics for the Analysis of Algorithms* by Greene and Knuth.
15. *Concentration of Measure for the Analysis of Randomized Algorithms* by Dubhashi and Panconesi.
16. *Handbook of Large Scale Random Networks* Edited by Bollobas, Kozma, and Miklos.
17. *Vehicular Networks: From Theory to Practice* Edited by Olariu and Weigle.

Books on Cryptography

1. *Introduction to Modern Cryptography* by Katz and Lindell.
2. *Concurrent Zero-Knowledge* by Alon Rosen.
3. *Introduction to cryptography: Principles and Applications* by Delfs and Knebl.
4. *Primality Testing and Integer Factorization in Public-Key Cryptography* by Yan
5. *Elliptic Curves: Number Theory and Cryptography* by Washington.

6. *Secure Key Establishment* by Choo.
7. *Algebraic Crypanlysis* by Bard
8. *An introduction to Mathematical Crytography* by Hoffstein, Pipher, Silverman.
9. *A Course in Number Theory and Cryptography* by Koblitz.
10. *Cryptanalytic Attacks on RSA* by Yan.

Books on Coding Theory

1. *Algebraic Function Fields and Codes* by Stichtenoth.
2. *Block Error-Correcting Codes: A Computational Primer* by Xambo-Descamps.
3. *Applied Algebra: Codes, Ciphers, and Discrete Algorithms* by Hardy, Richman, and Walker.
4. *Codes: An Introduction to Information Communication and Cryptography* by Biggs.

Books on Theory of Computation

1. *The Calculus of Computation: Decision Procedures with Applications to Verification* by Bradley and Manna.
2. *Computability of the Julia Sets* by Braverman and Yampolsky.
3. *Models of Computation: An introduction to Computability Theory* by Fernandez.

Combinatorics

1. *Applied Combinatorics* by Roberts and Tesman.
2. *Combinatorics the Rota Way* by Kung, Rota, and Yan.
3. *A Course in Enumeration* by Aigner.
4. *Chromatic Graph Theory* by Chatrang and Zhang.
5. *Design Theory* by Lindner and Rodger.
6. *Combinatorial Methods with computer applications* by Gross
7.
8. *A combinatorial approach to matrix theory and its application* by Brualdi and Cvetkovic.

Misc Books

1. *Quantum Computer Science: An Introduction* by Mermin.
2. *Complex Social Networks* by Vega-Redondo

3. *Branching Programs and Binary Decision Diagrams* by Wegener.
4. *When Least is Best: How Mathematicians Discovered many clever ways to make things as small (or as large) as possible* by Nahin.
5. *Stories about Maxima and Minima* by Tikhomirov.
6. *Decision and Elections: Explaining the Unexpected* by Saari.
7. *Creative Mathematics* by Wall
8. *Is Mathematics Inevitable? A Miscellany* Edited by Underwodd Dudley.
9. *Comprehensive Mathematics for Computer Scientists 1: Sets and numbers, graphs and algebra, logic and machines, linear geometry* by Mazzola, Milmeister, and Weissmann.
10. *Difference Equations: From Rabbits to Chaos* by Cull, Flahive, and Robson.
11. *Mathematical Tools for Data Mining* by Simovici and Djeraba.
12. *A Concise introduction to Data Compression* by Salomon.
13. *Practical Text Mining with Perl* by Roger Biliosly.
14. *The space and motion of communication agents* by Milner.

Review of[2]
Quantum Computation and Quantum Communication:
Theory and Experiments
Author of book: Mladen Pavicic
239 pages, Springer, $88.00
Author or Review: George Hacken

1 Introduction

Though post-relay and post-vacuum-tube digital computers are 'quantum' in the sense that Quantum Mechanics underlies solid-state (in particular, semiconductor) physics, Quantum Computing could, with my apologies to quantum field theorists, fairly be characterized as Computing's 'second quantization.' Pavicic's book achieves its own characteristic balance between the complementary attributes, depth and breadth, such that algorithmists and computation theorists will be reading both within and very much around their subject as that subject is conventionally construed. (James D. Watson advises in his recent book[3] that we should indeed "read around" our subject.) The book's preface states that "the theory of the field leads the experiments in a particular way .... As a consequence, both mathematics and physics are equally essential for any approach in [sic] the field and therefore for this book as well."

[2]©2009 George Hacken
[3]*Avoid Boring People*, Random House, 2007

Quantum Computation and Quantum Communication is subtitled *Theory and Experiments*; it comprises three chapters: Chapter 1, *Bits and Qubits*, is an explication of computation-theory as built up from, and related to, states that are quantum-mechanical ('coherent') superpositions of 0s and 1s, these latter being construed as quantum-mechanical Hilbert-space basis states (qubits) $|0>$ and $|1>$ that are subject to Cartesian-product juxtaposition for the case of higher dimensions. Chapter 2, *Experiments*, is a substantial enumeration of evidence for, and possibilities of, realizable quantum computers. Chapter 3, *Perspectives*, is a synthesis and elaboration of the first two chapters, ending in a section titled *Quantum Turing Machines vs. Quantum Algebra.*

As this review is at least *intended* specifically to serve the SIGACT community, I'll give my answer here to a thought-experimental question that I posed to myself after having read the book: "Quick! Is this a physics book or is it a computation-theory book?" My 'gut' answer, i.e., the eigenstate that the question ('measurement') forced me into was $|\text{Physics}>$. This is, of course, an incomplete characterization of the book, as the book gives ample evidence, via counterpoint and interleaving, of its author's and his colleagues' erudite mathematico-theoretical work in Computation Theory, which a good part of the book also reflects in a 'survey' sort of way with what I deem as pockets of mathematical depth sprinkled throughout and appearing especially at the end of the book.

I recall nodding in recognition when Donald Knuth wrote (somewhere and some time ago) that when he works in pure mathematics it has an entirely different 'feel' from that of computing, i.e., algorithmics *per se.* (For what it's worth, I have always found computing science to be impossible to distinguish from mathematics, but that's *my* problem.) Well, the pervasive physics parts of Pavicic's book, with their beam-splitters, polarizers, radiation by stimulated emission ..., gave me the ultimate physics-feel, i.e., *déjà vu*: Physics courses, even highly theoretical ones, generally come to grips with the non-Platonic reality of their subject, the infant String (a/k/a M) Theory to the contrary notwithstanding. Physics is inductive and, at large, does not *depend* on proof, but on evidence. This observation goes equally well for the theoretical-physics parts of Pavicic's book, in which P.A.M. Dirac's austere (but standard) bra-ket notation is used as a 'given.' The great Dirac is in fact quoted (elsewhere) as having said, "I am not interested in proofs, I am only interested in how Nature works," in response to accusations of a promiscuous lack of mathematical rigor. (The rigor was subsequently supplied by Laurent Schwartz, James Lighthill, et al; Dirac's physics remained invariant. Molière was also involved, as Dirac used group theory and Clifford algebra "without previous knowledge of [them]," to quote Dirac's response to an audience-member's challenge, "but you *are* using group theory!")

Pavicic states that the book "is not conceived as a textbook, ... but more as a guide." There is no homework, nor are there exercises, and "most of the required details are elaborated within the main body of the book." Pavicic intends for the reader to profit from a "polynomial complexity," once-through use of his book. The book is certainly not 'how-to,' but serves as an eye-opener, albeit a dizzying one for student and practitioner of 'classical' computing alike, to a genuinely new computing paradigm. The "classical" Turing machine is given enough detail, including formal definition of its deterministic species, to form the basis for at least a superficial notion of the quantum Turing machine's (currently tentative) role in the theory of quantum computation.

The beginning of the book gets right down to the business of computability by summarizing Turing-computability in terms of that abstract machine, and alluding to Church's conjecture that effective computability is what the theories of Turing, Church, Herbrand, Gödel, Kleene, Hilbert and Bernays, Post, and Markov have in common, i.e., are equivalent to. (Church's conjecture, a/k/a thesis is, of course, about the theory, not the theorists.) The important notion in addition to "calculability" (Pavicic's term) is decidability, and Pavicic achieves good cognitive and pedagogical flow to Boolean algebra by pointing out that "few proofs [of decidability and calculability] turned out to be possible for simple theories at the beginning of the twentieth century . . . ." In proffering the notion of axiom-based proofs, i.e., the theorem-proving or term-rewriting method over the brute-force truth-table-enumeration approach, the author makes the (I suppose) minor error that at least 30 billion truth-values need to be checked in verifying the correct functioning of a 16-bit adder that comprises over 30 boolean variables; the correct answer is more like one billion, as $2^{30} \approx 10^9$. I hasten to add that the prinicple remains illustrated, and that errors are the exception, not the rule, in this book. Pavicic is very clear throughout when he makes a scientific point, however nuanced computation theory *cum* quantum mechanics may be.

There is, in fact, a great deal of educational enrichment of *classical* computation-theory to be had here. Many of us know, for example, that the connectives *and, or, not* can be expressed via the *single* Sheffer stroke, |, which is the *nAnd* connective. The author quotes McCune et al's 2002 result to the effect that the *single* axiom $A|((B|A)|A))|(B|(C|A) \equiv B$ is equivalent to the five (depending on how one counts) axioms of Boolean algebra as based on *and, or, not*: closure; commutativity; identities 0 and 1; distributivity; and complements and inverses. (I don't intend to double-check *that*, as I suspect more than the back of an envelope, let alone skill, to be necessary.)

This axiomatic, abstract subsection is followed, in a fashion that is typical throughout the book, by an almost down-and-dirty physics section wherein bits realized by transistors are analyzed for energy dissipation, which is a major limiting factor in achieving speed of computation in physical machines. One major amelioration was industry's elimination of transistor-gate resistive losses via the synthesis of complementary metal-oxide semiconductor (CMOS) gates that are resistor-free (but capacitance-intensive). Pavicic goes on to mention more recent, post-CMOS, innovations it logic-gate technology, and thus segues into what I call an architectural, not technological, problem: irreversible gates. For example, in the Boolean equation $C = A \wedge B$, the 'culprit' in the 'output' C's being 0 (i.e., false) cannot be recovered when an ordinary, i.e., irreversible *and* gate processes inputs A, B. The section on the classical reversible versions of the Boolean operators[4] *not, and, or* prepares the reader for the quantum versions of reversible operations. This abstract, mathematical section is followed by its physics counterpoint, which here involves Gibbsian vector analysis, classical electromagnetic theory, and nonrelativistic quantum mechanics. Though I do not presume that SIGACT members at large have any less of a physics background than I do, I must state that this (and other) physics-intensive expositions are simply incomprehensible without such prior acquaintance; this can lead to a loss of patience on the reader's part, as the physics is anything but self-contained. For example (page 28) the notions of quantum-mechanical pure state versus that of a mixture (which latter is, in my words, doubly statisical), will be lost on the reader whose first exposure to quantum mechanics is this book. The same goes for the motivation behind

[4] I was lucky in this regard to have read Brian Hayes's, *Reverse Engineering*, American Scientist, March-April 2006

the introduction of hermitian and unitary operators (on quantum state-vectors). Of course, the beam-splitter as $\sqrt{NOT}$ gate will be a revelation to any interested party. Pavicic states the two main quantum-mechanical algorithms for classical applications to be factoring of integers (Shor) and search (Grove). A single ten-(decimal)-digit number needs only *one* q-gate, versus the several needed in current digital computing.

The section titled *Technological Candidates for Quantum Computers* is clear about the state of the art: "To date – about ten years after the first experimental implementation of one qubit – most of the numerous proposals for quantum computing prototypes have not been able to implement more than one or two qubits. Thus, one still cannot single out a most promising technological candidate for a future quantum computer." (Recall that, as alluded to above, a qubit $|q>$ is a superposition of basis states $|0>$ and $|1>$: $|q>= a|0> +b|1>, a, b$ complex with $|a|^2 + |b|^2 = 1$.) This is valuable real-world information for all of us, and I dare say that it is all the more credible by virtue of Pavicic's treatment of the physics that underlies quantum computing – however demanding or frustrating the reading of those parts of the book may be. Speaking of physics, I believe that the author is mistaken (in the typographical-error sense, on pages 112 and 117) in attributing *anti*commutation relations to Bose-Einstein (integer-spin) operators and commutation relations to Fermi-Dirac (half-integer spin) operators. Unless I'm missing something, it should be *vice versa*. The section titled *Future Experiments* begins with an enumeration of the DiVincenzo criteria, namely, "five generally accepted requirements for the implementation of quantum computing," plus two "networkability" conditions. The first is scalability, and the last is faithful transmission of qubits between specified locations. *Table 2.1*, page 124, shows how the following technologies currently fare with respect to these criteria: Nuclear Magnetic Resonance; Solid State; Trapped Ion; Cavity Quantum Electrodynamics; Neutral Atoms; Optical; Superconducting; and 'Unique' Qubits. None of them are currently viable for all seven DiVincenzo criteria. Near the end of Chapter 2 is mention of an existing, 18-mile quantum network administered by the Defense Advanced Research Projects Agency (DARPA), which includes quantum cryptography and anti-eavesdropping features.

Pavicic estimates, at the beginning of Chapter 3, *Perspectives*, that the shrinking of computer elements will encounter the quantum barrier by 2025. As quantum computing is "inherently parallel," he expects, by 2012, a 50-qubit computer that can accommodate a quantum superposition of 10^{15} states. The discussion on how 50 qubits evolve together to perform a quantum computation in one step is quite clear as a simple elucidation of the basic nature of quantum computing which, in view of the heterogeneous nature of this book, is a welcome summary. There is also a moderately detailed treatment of quantum communication *per se*, using the standard characters Alice (the transmitter or source), Bob (the receiver), and Eve (the eavesdropper).

The last sixth of the book is the most consistently mathematical and, if I may say so, SIGACT-like. It treats quantum algorithms, quantum algebra, and quantum Turing machines in a way that is closely parallel with their classical counterparts. Here we have our lattices, *C** and *Bear algebras*, and – yes – the Schrödinger equation as discretized for qubits. There is also treatment of a fascinating, purely quantum phenomenon, *counterfactual computation*, which involves non-destructive, interaction-free probing of a quantum state. (I learned this more than four decades ago as "the influence of the possible on the actual" in quantum mechanics; Pavicic tells me it's real!)

Quantum Computation and Quantum Communication is by no means an easy book, and there is no claim that it is. Its eclectic nature alone makes it demanding reading. That it captures its author's single, unified, and expert vision of quantum computing is a great strength. The book will be valuable for researchers, and for neophytes who want to get the 'flavor' of quantum computing, assuming that these latter beginners can keep their self-promises not to succumb to being overwhelmed.

2 Acknowledgement

We would like to thank Joshua Scott for help with the LaTeX.

Review of[5]

Quantum Computing for Computer Scientists
Author of Book: Noson S. Yanofsky and Mirco A. Mannucci
Cambridge University Press, 2008, 368 pages
ISBN-13: 978-0-521-87996-5, Price U$ 70.00

Review by S.C. Coutinho
Departamento de Ciência da Computação
Instituto de Matemática
Universidade Federal do Rio de Janeiro
P.O. Box 68530, 21945-970 Rio de Janeiro, RJ, Brazil.
`collier@impa.br`

1 Introduction

I first heard of quantum computing in 1995, a year after it had come of age. For more than ten years this had been a notion of only theoretical interest; then came Peter Shor's 1994 paper, and all was changed. What he proved was that a quantum computer would be able to factor integers and solve the discrete logarithm problem in *polynomial time*. Since the security of most internet communication depends on the difficulty of solving one of these two problems, his algorithms had a huge potential impact, and soon brought quantum computation into the news.

It was only much later that I learned that the roots of the subject go back to the 1980s, when Richard Feynman began to talk about simulating quantum mechanics with a computer. In a keynote address delivered at the MIT Physics of Computation Conference in 1981 he pointed out that Nature is not classical, and then added that

> if you want to make a simulation of Nature, you'd better make it quantum mechanical, and by golly it's a wonderful problem, because it doesn't look so easy; [1].

By 1983, Feynman was talking in terms of

> a computing device in which the numbers are represented by a row of atoms with each atom in either of the two states; [2].

[5]©2009 S.C. Coutinho

Another important development that was occurring at about the same time was the study of reversible computation by C. H. Bennett, E. Fredkin, T. Toffoli and others. Since quantum mechanical processes are always time reversible, quantum computation must use only reversible gates of the kind introduced by Fredkin and Toffoli.

The next major advance came from D. Deutsch. In a paper published in 1989 he showed that there exists such a thing as a *universal quantum computer*: a quantum computer that, like a universal Turing machine, can simulate any machine that works according to the laws of quantum mechanics. This discovery led Deutsch to propose that the well-known *Church-Turing principle* should be re-stated as

> there exists, or can be built, a universal quantum computer that can be programmed to perform any computational task that can be performed by any physical object.

With Peter Shor's 1994 paper we entered a new era in the study of quantum computers. Soon there were lots of people working on the subject, which led to new quantum algorithms for searching (Gover 1997) and quantum error correction (Shor 1995). By 1998 physicists were using nuclear magnetic resonance to create prototype quantum computers with a few qubits (or quantum bits). Since then there have been several advances, although the most impressive have probably been in *quantum cryptography.*

Although quantum cryptography is based on the same principles as its computer namesake, its technological requirements are more modest. Indeed, the first quantum cryptographic protocol was proposed as far back as 1984 by C. H. Bennet and G. Brassard — five years before Deutsch proved the existence of a quantum universal computing. Today there are at least four companies selling quantum cryptographic devices. These allow two parties to share a secret key that can then be used to encrypt and decrypt their messages. The advantage over conventional cryptography is that the principles of quantum mechanics make it possible to detect the presence of anyone trying to listen in order to get access to the shared key.

Recent successes in quantum cryptography seem to indicate that it has come to stay. However, the future of quantum computing is rather more uncertain. There are two main problems. The first is that we still do not have a quantum computer that can do anything really interesting. The second is that no new quantum algorithms, on a par with Shor's work, have been found for quite sometime. The main obstacle to the first problem is the well-known phenomenon of decoherence: when a quantum system interacts with the environment it collapses in an irreversible way that prevents the superposition of states required for a quantum computation to be effective. As to the second problem, no one really knows.

2 Review of the book

The book under review is an introduction to quantum computing and quantum cryptography. Although there are many other such books, this is the only one I know of that was written for computer scientists. This means, in particular, that the required background in physics is kept to a minimum. Thus, while most books on quantum computing begin with a fairly general introduction to quantum mechanics, Yanofsky and Mannucci managed to limit theirs to finite dimensional quantum systems. This also means that a solid knowledge of linear algebra will be enough for reading the book.

The book can be roughly divided into two parts. Chapters one to five form, what I will call, from now on, the *first part* of the book. This part begins with a very basic and elementary introduction to quantum mechanics. Actually, chapters one and two are a review of complex vector spaces and unitary transformations and may be omitted by those who already have a good knowledge of the subject. Chapter two also contains a section on tensor products, which are necessary for describing the superposition of states on which quantum computation depends. The leap into quantum theory, that comes in chapter three, is made smoother by a discussion of classical probabilistic systems that mimic some of the features of quantum systems. The axioms of quantum behaviour are introduced in chapter 4, while quantum bits (qubits) and gates are discussed in detail in chapter 5.

The second part opens with a description of some of the main quantum algorithms, including Grover's search and Shor's factoring. The next two chapters discuss programming languages and the computer science behind quantum computation. Chapter 9 is dedicated to quantum cryptography, and besides several special cryptographic protocols, it also contains a discussion of quantum teleportation. The last two chapters deal with quantum information theory, including error correction, and with possible hardware for implementing quantum computers. The book also contains five appendices. Two of these (A and D) are really a guide to the literature on the subject. There are also appendices with answers to selected exercises, experiments to be done with MATLAB and a collection of topics that may be used in student presentations.

3 Opinion

Given the hype around quantum computing, I was not surprised when undergraduates began asking questions about the subject. Although there are many articles and web sites that will give them a rough idea of what quantum computing is, most books on the subject — and there are several — begin with a fairly general introduction to quantum mechanics. Since most quantum systems are infinite dimensional, this presents a problem for computer science undergraduates, whose knowledge is often limited to finite dimensional vector spaces, and real ones at that. However, the quantum systems used in computing are all finite dimensional, and I kept hoping that one day someone would write a book that, taking this into account, would only require the knowledge of finite dimensional linear algebra that our students already have.

To my delight we now have just such an introduction, in the form of the book under review. Not only are vector spaces assumed to be finite dimensional throughout the book but, aware that most basic linear algebra courses deal only with real spaces, the authors included chapters on complex numbers and complex linear algebra. This makes for a truly elementary book that a computer science student, with a solid knowledge of vector spaces and linear transformations, should have no difficulty to read. Indeed, the authors are so careful in providing the right amount of detail that, to the more experienced student, this book will read almost like a novel.

This will also make it a very good textbook for an elementary course on quantum computing, although there is probably too much material to cover in one semester. Here the care with which the book was planned becomes apparent: as the graph of chapter dependencies on page xiv illustrates, chapters 6 to 11 are essentially independent of each other. The traditional course, which emphasizes the power of quantum over classical computation, corresponds to chapters one to six. However, a course on quantum cryptography can also be easily managed by moving directly to chapter 9, as soon as one has covered the first part.

The book contains many exercises (some with answers), most of which are rather elementary.

Actually, a few more challenging problems would have been welcome, especially in the first part. As one would expect from a book published by Cambridge University Press, this one is beautifully produced, and although there are some misprints, they are quite harmless.

To sum up, this is a book that I can recommend to anyone with a basic knowledge of linear algebra. Not only will it make a very nice textbook for undergraduate computer scientists and mathematicians; it is also the kind of book one can give to a bright student to read on her own.

References

[1] R. Feynman, *Simulating Physics with Computers*, Int. J. Theor. Phys. **21** (6/7) (1982), 467-488.

[2] R. Feynman, *Tiny Computers Obeying Quantum Mechanical Laws*, in New Directions in Physics: The Los Alamos 40th Anniversary Volume, Edited by N. Metropolis, D. M. Kerr, and G.-C. Rota, Academic Press, (1987), 7–25.

Review of[6] of
Biologically Inspired Algorithms for Financial Modelling
by Anthony Brabazon, Michael O'Neill
Springer-Verlag Berlin Heidelberg, 2006
275 pages, HARDCOVER

Review by
Brad G. Kyer, bkyer@acm.org

1 Introduction

Peering over the financial market landscape, it quickly becomes apparent that markets are dynamic and highly complex systems. The total number of variables that have a measurable effect on market valuation is immense and impractical for standard numerical techniques, especially when taken into context of real time market valuation. Additionally, market valuation may contain several variables with derived or perceived values (for example market sentiment indicators) which have no explicit discrete values but are implicitly represented in the final value. This is much like the cost associated to "risk"; the more risky an investment is the more an investor wants to earn above the risk free rate as compensation, but this varies between what the market implies as the risk compensation and what investors expect the compensation to be. Naturally, systems that are the fastest and can provide more accurate valuations in line with the market may provide an edge over traders using traditional or standard numerical valuations.

If one takes the approach that the market behaves much like a living, breathing, biological ecosystem then analogously biologically inspired algorithms may prove to be more effective tools. The objective of this book is to provide an introduction to biologically inspired algorithms and some tightly scoped practical examples in finance.

Many of the standard numerical techniques for valuation of more complicated financial markets like derivatives tend to be approximations, primarily for ease and performance based reasons. In

[6]©2008, Brad G. Kyer

addition, several numerical approximations used are based on numerical distributions that do not accurately reflect the real market [7] . There has been a significant number of works written on the subject in recent years, including the popular book "The Black Swan" [8]. These works focus on analyzing the system and the occurrence of events which current models deemed highly improbable (*this book has its roots in Mandelbrot's work*). If the recent debacles in the structured debt markets are any indication, the current definition of "improbable" needs to be re-evaluated.

2 Summary

The book is broken up into 3 sections with 21 chapters. The first chapter provides a basic introduction and the overall goal setting the authors wish to convey. The next 6 chapters are the "Methodologies" section, which give an overview of common biologically inspired algorithms that will be used in the case studies. The next 2 chapters are the "Model Development" section, where discrete goals, strategies and the mechanisms used for valuation are defined within the context of the author's goals. The last 12 chapters are the "Case Studies" section and present case studies using the techniques, algorithms and methodologies presented earlier in the book. The case studies tend to be brief, high level descriptions but vary on the amount of detail regarding the overall implementation and theory.

2.1 Chapter 1

The first chapter begins with an introduction of the goals of the book and a rationale of the similarities between financial markets and the biological systems. This is followed by cursory information about the biologically inspired algorithms to be utilized in the book to provide framing for what is to follow. The authors have selected several common biologically inspired algorithms, namely, Neural Networks, Genetic Algorithms, Grammatical Evolution, Particle Swarms, Ant Colony and Artificial Immune Systems as the building blocks for building the prediction systems. Finally, the authors provide basic but important information regarding scope for financial markets, the areas being modeled and how the results will be evaluated.

The introduction chapter is a quick read and provides a foundation for the following chapters of the book. Readers with little or no economics background will find the discussion of the Efficient Market Hypothesis and the Random Walk clear enough to understand related to what the biological system will be addressing, though many nuances regarding these tenets may require more independent research and analysis.

2.2 Chapter 2: Neural Network Methodologies

The second chapter dives right into the details of Artificial Neural Networks (ANN). This chapter is written for readers with no background in artificial neural networks and provides a working understanding of their behaviors, strengths and weaknesses. It is packed full of information, but readers who are not practitioners of artificial neural networks may need to re-read several sections

[7] "The (Mis) Behavior of Markets: A Fractal View of Risk, Ruin And Reward", Benoit B. Mandelbrot and Richard L. Hudson, Perseus Book Group

[8] The Black Swan: The Impact of the Highly Improbable, Nassim Nicholas Taleb, Random House

within this chapters (perhaps a couple times) to fully appreciate the information presented. To get a rough idea, there are nearly 21 pages devoted to artificial neural networks out of 256 total written pages in the book, so it is a fairly large percentage of the overall text. It is a crash course in artificial neural networks, however, so the depth of many of the discussions is necessarily limited and the user is expected to take them at face value.

For readers expecting to be able to go out and immediately implement one of the many styles of artificial neural network presented in this chapter (for example Self Organizing Maps); they may find that to be difficult, but that is not the true goal of the book.[9]

2.3 Chapter 3. Evolutionary Methodologies

The next chapter introduces Evolutionary Algorithms (EA). The main points of discussion are the Genetic Algorithm (GA), Differential Evolution (DE) and Genetic Programming (GP). They all share a common base feature set and the authors spend roughly 35 pages describing these methodologies. The first part of this chapter builds a wide breadth of information on Genetic Algorithms, making sure to discuss the core concepts and some more useful but complicated facets. The depth of discussion is at times cursory, but again, that's not the goal. Only a handful of pages are used to discuss the details of the somewhat more complicated Differential Evolution method but provide enough information to understand its principle and purpose. The last half of chapter 3 is dedicated to Genetic Programming and provides a wide breadth of information but with limited depth.

This chapter is crucial to understanding as it provides some of the basic functionality for Grammatical Evolution and will be used often case studies presented later. The authors discuss techniques for hybrid EA combined with Multi Layer Perceptrons (MLP, discussed in the Neural Network chapter) for evolving the structure of the MLP. The hybrid approach is powerful and utilizes complimentary features of both techniques to get better efficiency and performance.

2.4 Chapter 4. Grammatical Evolution

The fourth chapter provides a quick and thoughtful introduction into Grammatical Evolution methodologies. Using a powerful analogy, comparing a GE structure to that of Nature's DNA, the authors provide a working context for the framework. Facilitated by concepts from DNA (specifically genotypes, phenotypes and codons) one can easily make the next logical step toward envisioning evolving systems at meta-function levels. This differs from the lower level tightly encoded GA's or syntax trees like GP's. Grammatical Evolutionary methodology is one of the more favored tool sets by the authors for its inherent power and expressive ability [10] *(Note: "Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language" is a book also co-written by one of the authors of this book, Michael O'Neill).*

By inserting a mapping layer between the BNF grammar and the genome codons, evolving and mutating new sentences becomes possible while maintaining the implicit relation to the grammar definition. This is a valuable read into the abilities of the GE methodology, but this topic is deserving of a book unto itself. I suggest reading additional material to really get a deeper appreciation for what this is capable of (see previously mentioned book).

[9]Machine Learning, Tom Mitchell, McGraw-Hill

[10] "Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language", Michael O'Neil and, Conor Ryan, Springer

2.5 Chapter 5. The Particle Swarm Model" (PSO)

This chapter briefly touches upon a social learning model inspired by the flocking behaviors of creatures in nature (the origination is with birds). This is a powerful optimization technique, but the overall utilization by the financial industry as a whole has been limited, mainly due to the relatively young age of PSOs related to other methodologies and relative lack of information on usage criteria. The reader will find useful information and foundational knowledge in this chapter on PSO but the breadth and depth is unfortunately limited.

2.6 Chapter 6. Ant Colony Models" (ACO)

The sixth chapter introduces another social learning model, inspired by ants and their interactions within a colony. It is quite interesting and peculiar how ants operate without any centralized intelligence, yet as a collective group can perform amazing tasks. The authors limit their discussions to a subset of the Ant Colony Model metaphors called the ant-foraging model. This is common and popular model already used in network routing analysis and optimization. Though the chapter is relatively short it does contain a wide range of information regarding ant-foraging. Readers who want to learn more about this topic should read the seminal book titled "Ant Colony Optimization" by Dorigo and Stutzle[11].

2.7 Chapter 7. Artificial Immune Systems" (AIS)

The seventh chapter introduces another relatively new biological metaphor called the "Artificial Immune System" (AIS). This model attempts to emulate functions of the natural immune system for problem solving. The authors limit their discussions to the negative selection algorithm for classification and clonal expansion for optimization. There is a brief discussion on the natural immune system and how the AIS metaphorically would emulate this and provides a couple simple examples but more complicated examples are left to the reader to research. This is a very interesting technique and could prove to be quite powerful when applied as a negative indicator related to price movement expectations or model results. There is always the need for adaptable circuit breakers on automated trading systems when the trading environment no longer matches the model expectations.

2.8 Chapter 8. Model Development Process

Chapter 8 goes through the process for defining the prediction goals, the context for result values, how the result will be evaluated, and the input data mechanics. The authors begin by charting out a financial model work flow for prediction and offering different performance measure options with discussions for both pros and cons on each. The reader is strongly reminded to pay attention to details on data quality and time granularity of the data used in relation to the prediction expected. The effectiveness of evolutionary algorithms is directly impacted by poor quality data; it may focus the algorithm on irrelevant factors in the data.

This is an important chapter as it reintroduces financial concepts related to trading prediction and what are the appropriate ways to approach evaluating the results of the systems. The final goal of any prediction system for financial gain is to closely mimic reality and to make money by

[11] *"Ant Colony Optimization", Marco Dorigo and Thomas Stutzle, MIT Press*

being more exact then standard numerical approximations. It is paramount that one has to make sure their prediction system is firmly grounded in reality otherwise the fantasy of becoming rich will stay just that, a fantasy. Chapter 8 introduces several ways to evaluate the results but for the readers with limited financial backgrounds or lacking knowledge in regression techniques, may find some of these methods less clearly detailed. Additionally, this chapter does not get very deep into real life market rules and behaviors, simply mentioning a few of the more common ones. Any real implementation would need to take all the rules and behaviors into account to get a proper prediction engine. Again, this book is not geared to be a foot deep and an inch wide, rather it is more a foot wide and an inch deep with information.

2.9 Chapter 9. Technical Analysis

Systems that monitor and process market prices from the data feeds are designed to produce signals about the incoming data using a mix of historical, theoretical and live (current) pricing information. Technical analysis is a specialized field in finance that processes time series information to look for trends and (ir)regularities. This chapter does not propose to teach the reader to be a technical analyst, it simply discusses several common indicators, what the input data types needed and what the output produced means. The authors organize the indicators by style (moving average, momentum, breakout, stochastic and volume) which will make it easier for the reader to progress through and limit overall confusion. There are a significant number of indicators already developed [12] and documented by financial analysts and most institutions and analysts have specific preferences.

Additionally, the market behavior impacts what indicators make sense to use. A market moving horizontally will be more effectively traded, in general, with different indicators then a market exhibiting high volatility. This is not something discussed at length in the book, but then again, the book's primary goal is to provide ideas on what can be done, more then what to do when.

2.10 Chapters 10-21, The Case Studies

Many of the case studies presented in the book are higher level discussions then I would normally look for, as often the devil is hidden deep in the details, and these studies tend to be very similar in nature to IEEE or ACM style journal or research papers. Implementations are discussed at a high level with low detail and overall theory at a much lower level with high detail. This should not be viewed as a knock on the book, but for some of the more interesting chapters, I'd prefer to see more about the implementation and the case studies data made available, from either the authors or book publisher for further review. Many of the chapters are actually derived from scholastic papers and journal and that may be a limiting factor in some of the available information. For the most part it's adequate but there are some weak links.

Chapters 11 and 12 are primarily focused on using artificial neural networks for prediction. Chapter 11 is a straight forward neural network design using 1 network per indicator and combining all the different indicator results as inputs into another network which provides the final buy, sell or hold signal. Chapter 12 is slightly more complicated as it uses a GA to evolve both the network construction as well as the weighting between nodes. What is interesting is with GA evolved

[12]Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications (New York Institute of Finance), John J. Murphy, Prentice Hall Press

networks there may not be a single final answer for construction (though that begs the question on validation of the result).

Chapters 13-17, and 19 use Grammatical Evolutionary methodologies for prediction. Chapters 13 through 16 evolve predictive trading rules for different markets, chapter 17 evolves predictions on corporate failures and chapter 19 evolves bond rating rules. These chapters provide a wealth of approaches on meta-grammars using Grammatical Evolution. For the most part, many of these chapters do not provide discrete details about the implementation; rather they focus more on the grammar design and construction and the results generated by running the system. In order to keep the case studies simple, many of these implementations are based on a simplistic view of how the prices happen in a market. As is the standard for the book, determining what sort of other factors that should be used to further refine the system is an exercise left up to the reader. This is a valuable exercise for any reader who wishes to learn more about how all these systems behave in a real world scenarios.

As an example on how to enhance prediction on an equity market, one needs to look to other markets for additional information. In the case of a US equity market, the US bond market can provide indicators regarding expectations for short and long term interest and inflation rates and the foreign exchange (FX) markets for short and long term currency expectations. To understand how all this comes together, here's a simple but concrete example: On 11/23/2006 the S&P500 index (which is a basket of more than 500 equities) was up roughly 30% since 9/23/2001, however, the US dollar was down roughly 30% against the Euro currency. In absolute terms the S&P500 index was doing well, but in relative terms to the Euro dollar based economy it was simply breaking even. If you are an investor having to convert US dollars to Euro dollars, your investment in the S&P500 may not have been worth it; you could have simply held a Euro position and made 30% instead of nearly zero. Even as a US investor, the overall import prices have risen relative to the currency changes, making imported items more expensive.

Chapter 18 uses the Ant Colony methodology for predicting corporate failures by using the clustering model with a cemetery building technique. It is a rather novel approach for predicting corporate failures, as these failures tend to be more difficult to predict due to the sheer nature of the number of factors that can lead to failure under normal numerical techniques. The conceptual approach works in this example but may be confusing for readers who are new to Ant Colony Optimization techniques. Unfortunately, reading the text may not help clarify the confusion and many readers may need to reference another book on Ant Colony Optimization, previously mentioned.

Chapter 20 briefly describes implementing bond rating using AIS (artificial immune systems). The concepts of AIS are fairly deep and contain many semantics, which unfortunately, are not rigorously detailed in this text, though that is not necessarily a goal of the authors. Readers interested in more information regarding AIS should read the book "Artificial Immune Systems: A New Computation Intelligence Approach" by Leandro Nunes de Castro and Jonathan Timmis [13].

2.11 Chapter 21. Wrap-Up

This chapter reiterates the overall goals and objectives of the author's book, which is to provide an introduction and practical financial application examples with regards to biologically inspired algorithms. It also reminds the readers the often quoted standard financial warning "previous

[13] "Artificial Immune Systems: A New Computation Intelligence Approach", Leandro Nunes de Castro and Jonathan Timmis, Springer

success does not imply future success" still applies and that the reader is responsible to rigorously test their models before using them in the real markets. The key fact is that the modelling effort is never completely done, in the traditional sense. In order to use these, you have to evolve them on a periodic basis to consistently track the market trends.

3 Opinion

"Biologically Inspired Algorithms for Financial Modelling" provides some new insights and alternative tools for the financial modelling toolbox. As the financial market systems get more complex and efficient as more players enter the market, it becomes increasingly difficult to find an edge where a trader can make money using the standard tools and information available. The goal and objective of the book is to provide practical examples using these evolutionary algorithms and it does that decently, though I'd prefer to see more depth of information in the case studies. Most investors and firms look to optimize the ratio between amount of effort required and total positive cash flow as a relative value of the effectiveness. There is no economic sense to spend years modelling a system to make a few dollars.

If asked to build the case studies described using this book alone it would almost certainly be prone to failure, but that's not the point of the text; this book is meant as an idea and thought provoker then as a guide to implementation. Most readers will need more depth in information to actually construct many of the systems described, especially in regards to the Particle Swarms and the Artificial Immune Systems.

Overall I found the book very enlightening (and subsequently I went out and bought more related books) and it has provided ideas and alternative ways to think about solutions. I have implemented some of the biological algorithm case studies described by the authors, done some simple regression testing and found the results to provide useful insights. It will take a more thorough design, investigation and analysis, however, before these methodologies become a more mainstream approach.

Review of[14]

The Handbook of Bioinspired Algorithms and Applications
Book Edited by Stephan Olariu and Albert Y. Zomaya
Chapman & Hall / CRC, 679 pages, $139.95
Reviewed by Kushal Chakrabarti (kushalc@amazon.com)

Overview

The Handbook of Bioinspired Algorithms and Applications [4] seeks to provide, with little success, a starting point for researchers wanting to apply biologically-motivated algorithms to their specific problems. Although impressive in breadth — it contains articles on everything from neural networks to cellular automata to distributed systems design — the book does not provide enough depth or rigor to be useful to practitioners in the field. Due to its organization as a handbook and the sheer number of contributors — 85 authors from 17 different countries! — many sections of the book are redundant, and the quality of chapters varies widely from virtually unreadable to satisfyingly informative. All in all, with significant restructuring and editing, this book has the potential to

[14]©2009 Kushal Chakrabarti

be very useful to researchers in many disciplines. Until then, readers may find that an in-depth reading of freely available review articles and resources [1, 2, 3, 5, 6, 7] is probably a better use of their time and money.

Summary

The book is divided into two primary sections: descriptions of the core algorithms and heuristics themselves, and applications of those algorithms to specific problems. Several chapters from the first section have been summarized and reviewed in depth below, but I have elected to omit[15] summaries and reviews for the remaining few: Parallel Cellular Algorithms and Programs, Decentralized Cellular Evolutionary Algorithms, Optimization via Gene Expression Algorithms, and Dynamic Updating DNA Computing Algorithms. The second section contains 25 chapters describing applications of ideas from the first section; a few have been reviewed below, but, due to space constraints, the majority have been left out. To give some idea of the breadth of this second section, however, I include a sampling of chapter titles:

- Minimization of SADMs in Unidirectional SONET/WDM Rings Using Genetic Algorithms;
- Scheduling and Rescheduling with Use of Cellular Automata;
- Simulating the Strategic Adaptation of Organizations Using OrgSwarm;
- Biological Inspired Based Intrusion Detection Models for Mobile Telecommunication Systems;
- Bio-Inspired Data Mining;
- Solving the Partitioning Problem in Distributed Virtual Environment Systems Using Evolutive Algorithms; and,
- Frameworks for the Design of Reusable Parallel and Distributed Metaheuristics.

Instead of sequential order by chapter, I have elected to organize the review along subject lines, which I hope will improve clarity and ease comparison between related chapters. Although I have intermingled specific comments along with summaries wherever appropriate, a concluding general review follows in a separate section after the summaries.

Evolutionary Algorithms

Chapter 1: Evolutionary Algorithms (E. Alba and C. Cotta)

Evolutionary computation and, in particular, genetic programming, often provide the classic examples of biologically inspired computational techniques. The fundamental idea underlying these algorithms is their generation and iterative improvement [i.e. evolution] of a collection of possible solutions [i.e. a population] to the original problem until some quality requirement is satisfied or resource constraints are exceeded. As described by a reference in the article,

> The algorithm maintains a collection of potential solutions to a problem. Some of these possible solutions are used to create new potential solution through the use of operators. [...] The potential solutions that an operator acts on are selected on the basis of their quality [...]. The algorithm uses this process repeatedly to generate new collections of potential solutions until some stopping criterion is met.

[15] Given the referential nature of a handbook, this summary is by no means exhaustive. Instead of superficially summarizing the whole book, I have instead tried to provide useful summaries and commentaries on a small selection of chapters. It is my hope that the selection stands as a representative sampling of the book's content and quality.

Ultimately, different evolutionary algorithms and implementations differ in how (a) quality is defined, (b) solutions are selected for replication, given a definition of quality, and (c) new solutions are proposed from existing ones that have been selected. All this closely follows the biological metaphor: there, given an existing population, competition for finite resources favors the fittest individuals, allowing them to proliferate [compare to (c)], but, during sexual reproduction, normal variation-inducing mechanisms [e.g. chromosome recombination] produce slight differences in future generations [compare to (b)].

The article by Alba and Cotta provides this same general background, which, although all available, is scattered about six pages and buried amongst other extraneous information. Specifically, for a book about biologically inspired algorithms, in an introduction to the dozen or so chapters dedicated to evolutionary algorithms, it is odd that there is only half a page dedicated to the biological underpinnings of these methods. Some of the notation introduced in this chapter is rather confusing [Figure 1.1], and, for better or worse, is not used in the remainder of the chapter — let alone book. What I found particularly disappointing, however, was their summary of applications of evolutionary algorithms: instead of discussing the empirical tradeoffs of evolutionary algorithms relative to, say, Monte Carlo algorithms, they simply laundry list more than two dozen problems and areas in which they have been applied. Although the reader could mine the 51 references cited in the five paragraphs there, and tabulate the advantages and disadvantages himself, no reader ever will. Evolutionary algorithms are very interesting, understanding how their capabilities compare to the various other optimization algorithms would have been very useful.

Despite a mediocre beginning and end, however, Alba and Cotta shine in their dissection of the evolutionary algorithm framework. They provide significant detail on the various components, discussing the design of good definitions of quality (fitness functions), tradeoffs for population initialization and selection strategies, and many different approaches to generate new proposals. For example, in discussing strategies to select individuals for replication,

> The most popular techniques are fitness-proportionate methods. In these methods, the probability of selecting an individual for breeding is proportional to its fitness, that is,
>
> $$p_i = \frac{f_i}{\sum_{j \in P} f_j}$$
>
> [...] Another problem [with fitness-proportionate methods] is the appearance of an individual whose fitness is much better than the remaining individuals. Such *super-individuals* can quickly take over the population. To avoid this, the best option is using a nonfitness-proportionate mechanism. A first possibility is *ranking selection*: individuals are ranked according to fitness [...], and later selected [...] using the following probabilities
>
> $$p_i = \frac{1}{|P|}[\lambda^- + (\lambda^+ - \lambda^-)\frac{i-1}{|P|-1}$$

The approach to generating new proposals in evolutionary algorithms warrants some attention. What is particularly interesting here is that, unlike Monte Carlo algorithms that store only one solution at a given time, evolutionary algorithms can exploit their population of solutions here: by finding good partial solutions in different individuals and combining them, it can generate an individual solution that is better than its constituent parts. This, in fact, occurs in both biological evolution [e.g. crossover events during meiosis] and in evolutionary algorithms [e.g. a bitmask combining two bitstrings] during a step called recombination. Like random point mutations in DNA in biological evolution, evolutionary algorithms have a mutation step in which a neighboring solution is stochastically generated from an existing one [e.g. a random bitflip in a bitstring]. Although Monte Carlo algorithms have an analogous step, the recombination step is unique to evolutionary algorithms and could possibly be exploited to speed convergence relative to Monte Carlo. Unfortunately, as mentioned before, Alba and Cotta omit any such comparative discussions.

Although the discussion here covers a lot of ground, it, unlike the remainder of the section, is concrete enough and deep enough to leave the reader with a reasonably good intuition of the landscape of evolutionary algorithms. Had Alba and Cotta also provided a similar description of how evolutionary algorithms fit into the landscape of optimization algorithms, this chapter would have been quite rewarding.

Chapter 5: Parallel Genetic Programming (F. de Vega)

Genetic programming is a specific class of evolutionary algorithms in which the population consists of computer programs usually encoded as simple parse trees. As elsewhere, these parse trees contain functions and operators [e.g. if-then-else, addition] as internal nodes and inputs and constants as terminal nodes. Here, recombination often occurs through the probabilistic replacement of an entire subtree from a parent and mutation through the direct probabilistic modification of an existing subtree.

Although the introduction to genetic programming was quite good in and of itself, in relation to the previous chapter, nearly all of it was redundant. The primary differences between the two introductions has been highlighted in the previous paragraph, requiring just three sentences — the original description required five pages. This lack of concision seems to be a general problem throughout the book.

To this introduction to genetic programming, de Vega adds a description of parallelized genetic programming, which, as it turns out, provides not only computational benefits but also algorithmic improvements. Most interestingly, one can parallelize genetic programming by partitioning up the population into semi-independent sub-populations, and allowing different processors to simulate the different sub-populations. In this island model of parallelization, these different sub-populations (demes) are independent of one another except for an occasional stochastic exchange of individuals, which of course adds several new parameters about the number of demes, frequency of exchange, communication topology, etc. In addition to the potential computational benefits of parallelization, this approach has also been shown to improve convergence by increasing diversity and allowing the simultaneous exploration of multiple sub-spaces.

Moving past the initial description, de Vega attempts to illustrate the advantages of parallelized genetic programming on two "real-life" examples: automatic synthesis of FPGA circuits, and diagnosis of burn severity. Although significant background information is provided about FPGAs and circuits, virtually no information is provided about the specific problem being solved or how it is being solved. Furthermore, a plot without any meaningful labels or confidence intervals is used to declare that "parallel GP, employing 5 populations and 500 individuals per population, achieved better convergence results than GP with 2500 individuals."

Although burn severity diagnosis is far better described, there are similar problems with its conclusion: given an essentially unlabeled plot and no other analysis, de Vega declares that "parallel GP employing 5 populations and 2000 individuals per population achieved better convergence results than GP with 10000 individuals." This example, however, seems to contain a far more egregious flaw: the dataset used to train the algorithm is the same dataset used to evaluate it. Quoting from the article,

> In order to train the system, a table of parameters, taken by studying the photographs and diagnoses, was built. A set of 31 clinical cases were provided by doctors, and these cases were used to train the system. Each of them was allocated its corresponding parameters together with the result of its evolution over a couple of weeks. This table of values was given to the algorithm for it to train.
>
> [...]

> We ran the algorithm employing 5 populations of 2000 individuals each, with a period of migration of 10 generations. We waited for 60 generations before taking the results. At the end, the best decision tree we obtained was able to classify correctly 27 out of 31 cases.

Assuming this is not just an error of explanation, this invalidates the entire example: if the data used to evaluate an classification algorithm is the same as the data used to train it, the results of that evaluation do not reflect the performance of the algorithm on previously unseen data.

What is even more odd about this example is just how small it is: the algorithm considers just three features of burn data, and the total dimensionality of the feature space is 20. For each of the 20 possible configurations of the three features, there are four possible diagnoses, meaning that a naive brute-force search need only investigate 80 possibilities to find the globally optimal solution. Given the vague labeling of the provided plot — what does an effort of 600000 mean? — it is difficult to know whether this was more or less computational effort than parallelized genetic programming. The brute-force solution, however, is inarguably simpler and guaranteed to both terminate after exactly 80 iterations and guaranteed to find the optimal solution. Why bother with a genetic algorithm, parallelized or unparallelized?

Excluding the discussion of the potential algorithmic benefits of parallelized genetic programming, this chapter was thoroughly disappointing. It simply just does not provide much useful information, due to the aforementioned poor evaluations and redundant descriptions. The chapter promises real-life applications — not only is it part of the title, but the chapter begins by suggesting that, due to the high costs of hiring software engineers, genetic programming may be useful for automatic industrial software development — but fails to live up to that promise.

Chapter 12: Setting Parameter Values for Parallel Genetic Algorithms (M. Moore)

As mentioned above, a number of free parameters (e.g. number of sub-populations) exist for parallel genetic algorithms. Unsurprisingly, the success of a particular genetic algorithm can strongly depend on the choice of these parameters, and some recent effort has been dedicated to making the design of parallel genetic algorithms less of an "empirical black art." In this article, Moore argues that, although automatic parameterization methods may work in some restricted cases, these methods may not yet model parameters in the general case.

The specific problem used by this article to evaluate these parameterization methods is task scheduling on a multi-processor cluster: given N tasks with known processing and communication costs, the genetic algorithm must generate a schedule distributing these tasks across M different processors so that the total time to complete all the tasks (the makespan) is minimized. In Moore's design,

1. each solution[16] encodes processor assignments for each task in an M-ary string of length N;
2. selection occurs by discarding all individuals that fall below the average fitness in their local deme, importing the most fit individuals from other demes, and repeatedly mating the surviving individuals until the open positions have been filled;
3. reproduction occurs through recombination, in which the child randomly selects each processor assignment from its parents, and mutation, in which the child's assignments are randomly regenerated with probability 0.025; and,

[16]This solution seems to assume that the tasks cannot be reordered, which is a constraint not generally placed on schedulers. Moore even states that "[d]ependent computations are grouped into a single task, so that each task is independent" which enables exactly this kind of reordering. Although odd, it is part of the problem formulation, and no specific fault can be made.

4. the final answer of the scheduler is the best schedule found in 1000 generations, whose makespan is compared to the optimal makespan.

The main parameter remaining free in this design is the size N_d of each deme, for which Moore derives multiple fits according to different levels of expected optimality. Although I omit some details here for simplicity, she derives that

$$N_d = \frac{v\sqrt{-M \ln (1 - cQ)}}{2v - 1}$$

where v is a function of fitness variance, c is a constant depending on the number of processors, and Q is the desired solution optimality. For different values of Q, Moore then estimates N_d and determines the empirical optimality for normally and exponentially distributed task costs. For normally distributed data, the parameter fits seem to be reasonably accurate, although the predicted optimalities are slightly [under 10%] but consistently biased below the empirical ones. With the exponentially distributed data, however, the predicted and empirical optimalities diverge widely [over 40%] and are quite disappointing. Unfortunately, because Moore applies a scaling factor to the original derivation to correct for some "excess quality" at lower predicted optimalities,

> This "excess" quality was not a problem in practical terms; however, it indicated that the sizing equations were not capturing an important feature in the behavior of the scheduler. [...] Since the deme sizes produced accurate results at the higher range, a scaling factor was needed to make the deme sizes reflect the rate of increase at the lower quality levels. Deme sizes were scaled so that they used the calculated size for the highest confidence level. These sizes were decreased at an exponential rate until they reached the lowest confidence level.

it is unclear how much of the divergence is due to the parameter fitting methods and how much is due to the scaling factor. Considering that the predicted and empirical optimalities are actually quite similar for higher values, and that the divergence appears only at the lower values where Moore's scaling would have had the greatest effect, the errors may be primarily due to this additional confounding factor.

Although there may be technical problems with Moore's analysis, this article is still one of the better chapters in the book. It provides a strong introduction and motivation and a clear and concrete explanation of the problem and solution design, which is what one would expect from an example application chapter in this book. I do have some concerns over the technical accuracy of her evaluation, but it at least provides a good case study of how one could use apply parallel genetic algorithms to a specific problem and use automatic parameter fitting methods.

Swarm Intelligence

Chapter 3: Ant Colony Optimization (M. Guntsch and J. Branke)

One remarkable fact in insect colonies is the emergence of complex social cooperation [e.g. discovery of short paths during food foraging] in the interactions of very simple agents [e.g. in ants]. Dubbed swarm intelligence in computer science, analogous techniques have been used for classic optimization problems such as the Traveling Salesman Problem (TSP) and shown to be among the best for certain restricted cases. In arguably the best chapter of this book, Guntsch and Branke (a) present very interesting and accessible biological background about Ant Colony Optimization (ACO), a specific case of swarm intelligence, (b) describe an intuitive and clear application of ACO to the TSP, (c) compare and contrast different specific design decisions for applications to other problems, and (d) survey various applications of ACO, discussing both successes and failures.

In investigating how ants discovered shortest paths, biologists in the late 1980s discovered that ants probabilistically chose between paths based on the relative strength of pheromones left behind by earlier visitors. New visitors prefer paths with stronger pheromone concentrations, and, over time, as the pheromones evaporate, unvisited paths receive no new pheromones and thus very few visitors. Although it is clear that this can be made to converge to a particular path, what isn't obvious is whether or why this converges to the optimal path. I omit the specific discussion here for brevity, but Guntsch and Branke provide an intuitive example of the mechanics in a small closed loop setting.

The application of ACO to the TSP is direct in the sense that the computer science application is a small generalization of the natural phenomenon searching for short paths between the food source and nest. Given an edge-weighted graph $G = (V, V \times V, d)$ with $d : V \times V \to \mathcal{R}^+$ indicating distances between nodes, the algorithm maintains two matrices: τ_{ij}, which encodes pheromone information about the historic goodness of jumping from city i to city j, and $\eta_{ij} = d_{ij}^{-1}$, which encodes information about the immediate goodness of jumping from city i to city j. At every step of the algorithm, it iterates through each ant, which builds its tour by (a) starting from a random city and (b) probabilistically selecting its next city j from the set S of cities not yet visited with probability

$$p_{ij} = \frac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{h \in S} \tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}$$

for weights $\alpha, \beta \in \mathcal{R}^+$. Once all the ants have completed their tours in a given iteration, the algorithm (a) exponentially decays all earlier pheromone values and (b) increments pheromone values for the best tour in the current iteration and over all iterations. More precisely, (a) the earlier pheromones are first decayed according to $\tau_{ij} \mapsto (1-\rho)\tau_{ij}$ given an evaporation rate $0 \leq \rho \leq 1$, and (b) the pheromones for each city-jump (i, j) in the two best tours are then incremented according to $\tau_{ij} \mapsto \tau_{ij} + \frac{\Delta}{2}$. As an example of their attention to detail, Guntsch and Branke report here that specific values of $\alpha = 1$, $\beta = 5$, and $\Delta = \rho$ work well for this application.

With this specific example of ACO, Guntsch and Branke then move onto a detailed discussion of different (a) interpretations of the matrix τ_{ij}, (b) approaches to construct solutions from the matrices τ_{ij} and η_{ij} for specific ants, (c) methods for updating the pheromone matrix, and (d) initializations of the pheromone matrix. Although completely relevant and extremely informative, I again omit the discussion for brevity and instead only provide an example. From their discussion of different pheromone update strategies,

> Formally, an update takes the form $\tau_{ij} \mapsto (1-\rho)\tau_{ij} + \Delta_{ij}$ where $\rho \in (0, 1]$ is a parameter of the algorithm denoting how much of the pheromone information is lost with each application of evaporation. A high evaporation rate will cause a more rapid convergence which is coupled with less exploration than a low evaporation rate. [...] Δ_{ij} is an update value, which is 0 if the edge (i, j) was not traversed by the ant and some value greater than 0 if it was. The exact value of Δ_{ij} and especially the strategy when an update is performed is the key difference between most types of ACO algorithm[s].
>
> [...]
>
> Three different methods of determining the individual Δ_{ij} were tested: assigning a constant, using the inverse to the distance d_{ij} between [cities] i and j, and, performing best and used subsequently, inverse to the length of the entire tour, that is, the solution quality. [... I]t was also proposed to let a number of so called elitist ants, which represent the best solution found by all ants so far, update the trail. Using a small number of these ants, inspired by the elitist strategy in Reference 28, intensifies the search near the currently best solution, leading to better results overall.

The authors conclude by surveying "some of the noteworthy applications of ACO," which stands far ahead of the other surveys in this book by not only listing those noteworthy applications but also discussing their merits relative to competing approaches. In discussing the TSP, for example, they note that although it was a great first demonstration of ACO and produces good results, better non-ACO-based approximations exist. On the other hand, it turns out that ACO is one of the best known solutions for discovering minimum-weight Hamiltonian paths in graphs with precedence constraints (Sequential Ordering Problem).

As I mentioned previously, this chapter is arguably one of the best in the book and one of its redeeming factors. Not only was it a pleasure to read, it was also a very informative and thorough introduction to ant colony optimization. For a book whose goal is provide "a repository of how-to-apply bio-inspired paradigms," this chapter informs the reader of not only how to apply ant colony techniques but also where to apply them. At the end of the day, I can say that, after just one or two readings of this well-written chapter, I have learned enough about ant colony techniques to begin applying them in my own problem domains.

Chapter 4: Swarm Intelligence (M. Belal et al)

Generalizing the previous chapter on ant colony optimization, this chapter attempts to abstract out some higher-level concepts in swarm intelligence techniques. It notes that such techniques operate in an intrinsically bottom-up approach, with complex behavior emerging in a large population of independent entities programmed with simple rules, and states some of the "organizing principles" and "paradigms" behind these approaches. Unfortunately, this chapter turns out to add little value beyond that of the previous chapter, and to be too abstract and too vague to be of any real use to researchers in the field.

Belal *et al* declare that the "[swarm intelligence] approach emphasizes two important paradigms: the highly distributed control paradigm and the emergent strategy-based paradigm." Although certainly true, the actual explanation of these paradigms is too vague to pack any informative punch. For example, in the section describing the "emergent strategy-based paradigm," the authors give an example of model ants creating clusters of similar objects from an initial random distribution of objects. Although the phenomena is very interesting, the authors provide only the following justification and with neither any high-level intuition or lower-level mathematical grounding:

> In fact, such interesting collective behavior can be mediated by nothing more than similar, simple individual behavior. For example, F(1) (sic) each ant wanders a bit, (2) if an ant meets an object and if it does not carry one, it takes it, and (3) if an ant transports an object and there is a similar object in the same way in front of it, it deposits its load.

That global sorting emerges from such local behavior is imaginable, but the reasons for its emergence or limitations of such emergence are not at all described. To base the remainder of this chapter and the other swarm-intelligence-based applications in the book on just this imagination without any further formalization seems like a rather delicate proposition.

Ignoring a redundant explanation of indirect communication and ant colony optimization, the authors then get into another main specific case of swarm intelligence: particle swarm optimization. In this technique, which was inspired by the bird flocking and fish schooling behavior, a population of particles fly around in a multi-dimensional search space and, at each iteration, updates its position "according to its own experience and [the] experience of its of a neighboring particle." In particular,

$$\begin{aligned} v_i(t+1) &= av_i(t) + b_i r_i(t)(q_i(t) - p_i(t)) + b_s r_s(t)(q(t) - p_i(t)) \\ p_i(t+1) &= p_i(t) + v_i(t+1) \end{aligned}$$

where $p_i(t)$ and $v_i(t)$ respectively denote the position and velocity of particle i at time t, $q_i(t)$ the historical best position of particle i up to time t, and $q(t)$ the historical best position of the neighborhood up to time t; in addition, a, b_i and b_s denote free tuning parameters to weight the respective contributions of the previous velocity, velocity to reorient a particle i to its historically optimal direction, and velocity to reorient all particles to the swarm's historically optimal direction; finally, $r_i(t), r_s(t) \in [0, 1]$ are random numbers used to induce wider exploration of the search space. Given this definition of particle swarm optimization, Belal *et al* give an application to task assignment in a computational cluster, for which the only explanation follows:

> [E]ach particle is represented by a vector of length equal to M, which corresponds to the number of vertices in the task-graph. The value of each element in the particle is an integer in the range $[1, N]$, where N is the number of processors in the target architecture. [...] Thus, a particle corresponds to an M-coordinate position in an M-dimensional (sic) search space.

Here, both the description of the particle swarm equations and the proposed solution to task assignment problem are confusing[17] and only mechanistically explained without any intuition of why this technique works or where it works. These problems, in addition to the general imprecision and redundancy of the chapter, prevents it from supplying much useful information beyond the previous one on ant colony optimization.

Chapter 22: Swarming Agents for Decentralized Clustering in Spatial Data (G. Folino et al)

This chapter describes the extraction of simple heuristic rules from traditional clustering techniques and their incorporation into swarming agents for the clustering of spatial data. It contains a excellent introduction and a reasonable description of the traditional and swarm-based algorithms, but starts to degrade near the end. Although the chapter seemed initially promising, the evaluation of their ideas is very poor and it's not at all clear that their algorithms are as good as or better than their baseline heuristics. As a passing remark, to the extent that the authors seem to be proposing a new algorithm, this chapter should really be published through peer review for technical comments and verification – not as a chapter in a book.

Folino *et al* start by pointing out that, although many different approaches to clustering have been investigated, all rely on a centralized core. With the availability of very large datasets and distributed computing systems, he rightly argues that decentralized algorithms based on swarm intelligence could be very useful. To this end, they extract from DBSCAN, a simple spatial clustering algorithm, the concept of core points, which are points that have some minimum number of points within a specified neighborhood. They combine this heuristic with a model of bird flocks to produce the SPARROW algorithm:

1. Randomly disperse initial agents. Depending on the number N of neighboring points, choose a color: red $[N > M]$, green $[\frac{M}{4} < N \leq M]$, yellow $[0 < N \leq \frac{M}{4}]$, or white $[N = 0]$ where M is the minimum number of neighboring points described above for DBSCAN.

2. For every yellow or white agent, compute a direction away from white agents and toward red agents. Move in this direction with velocity 2 if yellow and velocity 1 if green.

3. For every white agent, generate a new agent that is randomly dispersed in the entire space.

4. For every red agent, merge together the neighboring cluster and also generate a new agent that is randomly dispersed in the near vicinity.

[17] The description is confusing to the point of actually being incorrect in the task-assignment model. There, as quoted above, the dimensionality of the search space is actually the product of the number of processors and tasks, i.e. $M \times N$, not just the number of processors M.

5. Regenerate and disperse agents whose age exceeds some threshold.

Here, colors are used to indicate areas of (dis-)interest. Just as white flags a zone of negative interest, yellow indicates a zone of less interest than green, and hence the agent moves faster out of it. It turns out that, occasionally, a non-white agent can become trapped by white agents, so, after some period of usefulness, agents are regenerated elsewhere. Unfortunately, neither DBSCAN and SPARROW cannot handle clusters of different densities, so the authors incorporate heuristics from the SNN clustering algorithm to make SPARROW-SNN. I have omitted the details for brevity.

Ironically, although Folino *et al* motivate their approach by suggesting that decentralized clustering algorithms are necessary in a world of large datasets and distributed computing systems, their algorithm (a) only performs better than DBSCAN, their baseline heuristic, for very small fractions of the original dataset and (b) is not currently parallelized. Furthermore, their evaluation does not (c) compare against the state-of-the-art in clustering algorithms, or (d), in the case of SPARROW-SNN, compare even against some non-swarming algorithm. Finally, the authors measure the quality of their clustering by the number of core points, and (e) it is not clear that a greater number of core points indicates a better clustering result. Ultimately, for the purposes of this book, this chapter may illustrate how one may translate traditional algorithms into swarming-based algorithms but it fails the reader on many other accounts.

Frameworks and Foundations

Chapter 10: A Unified View on Metaheuristics and Their Hybridization (J. Branke et al)

In Chapter 10, Branke *et al* promise a "simple unified framework [describing] the fundamental principle common to all metaheuristics" that, "[d]ue to its simplicity and generality, [suggests] a natural way for hybridization." They deliver a characterless and sweeping plain description of iterative search algorithms. While useful, I would consider it a basic explanation of the algorithms – nothing particularly novel – and it by no means lives up to its introductory hype.

¿From the section entitled "A Unified Framework for Iterative Search Heuristics,"

> The proposed general model [...] is depicted in Figure 10.1: new solutions are constructed based on information stored in the memory, possibly involving several construction operations that may be applied in parallel (e.g. [...]) or sequentially (e.g. [...]). The construction operators can be rather simple (as e.g. a single bit flip) or rather complicated (e.g. a local optimizer). The new solutions are then evaluated and can be used to update the memory, after which the cycle repeats.

and a following section in which the above "general framework" is used to describe standard metaheuristics,

> Evolutionary algorithms store information about the previous search in the form of a set of solutions (population). New solutions are constructed by selecting two solutions (parents), combining them in some way (crossover), and performing some local modifications (mutation). Then, the memory is updated by inserting the new solutions into the population. Although there exist a variety of evolutionary algorithm variants, they all fit into this general framework. For example, evolution strategies with self-adaptive mutation can be specified by extending the memory to also maintain information about the strategy parameters. Steady state genetic algorithms update the population after every newly generated solution, while genetic algorithms with generational reproduction generate a whole new population of individuals before updating the memory.

Although this is a good explanation of evolutionary algorithms – better than some others in this book – it does not particularly benefit from the so-called framework. Frameworks are useful when they abstract significant mechanistic components from its constituents, illuminating the core concepts, or generalize existing ideas to suggest new ones. It is true that this framework does generalize iterative search algorithms, but it does so in the most trivial of ways, without any explanative power or benefit.

The novel hybridizations suggested in the introduction ultimately end up being relatively simple combinations of standard ant colony optimization and evolutionary algorithms. The simplest fusion algorithm consists of generating 50% of solutions through ant colony optimization and 50% through evolutionary algorithms. In all fairness to the authors, however, I also describe a less trivial fusion from the fusions: pheromone-supported edge recombination (PsER) crossover. In this hybridization for the Euclidian TSP, if the evolutionary algorithm ever attempts to visit cities that have already been visited, the city is probabilistically selected from the pheromone matrix.

The authors then evaluate the standard evolutionary algorithm, the standard ant colony optimizer described previously in this review, a crippled ant colony optimizer from which edge-weight distance information has been withheld, and five hybrids. Among the hybrids without edge-weight information, PsER performs the best, and the standard ant colony optimizer and another hybrid perform slightly better. What I found most interesting about this evaluation, however, were the algorithms' performances on cities distributed about the unit circle. Here, the optimal tour length is clearly 2π, but of all the algorithms presented, only three were able to discover this optimal solution and another three had errors of more than 95%. Metaheuristics are very interesting, but, as I have mentioned multiple times in this review, rigorous evaluations are absolutely necessary to determine the scientific worth of specific algorithms. This book seems to have a systematic problem on this point.

Chapter 11: The Foundations of Autonomic Computing (S. Hariri et al)

Although the book advertises this chapter as describing "the foundations of autonomic computing, a revolutionary paradigm in modern complex networks," it suffers from the same problems as the previous chapter: it describes a framework that, while interesting, lacks any clarifying power. Drawing an analogy to the autonomic nervous system, which is responsible for maintaining homeostasis in higher-order animals, the authors essentially describe the concept of negative feedback and show that it exists in a number of experimental distributed systems. The concept of autonomic computing is indeed very powerful – in fact, a recent issue of the Operating Systems Review from ACM SIGOPS happens to focus exclusively on self-organizing systems – but this chapter does not do it justice.

In trying to abstract out principles of autonomic computing, Hariri *et al* draw a parallel to the biological autonomic nervous system, and explain how it can be modeled via a model of adaptive behavior proposed in 1960 by Ashby. They then take this model of adaptive behavior, apply it to autonomic systems, and declare that any "ultrastable system architecture" must have

> (1) sensor channels to sense the changes in the internal and external environment, and (2) motor channels to react to and counter the effects of the changes in the environment by changing the system and maintaining equilibrium. The changes sensed by the sensor channels have to be *analyzed* to determine if any of the essential variables has gone out of their viability limits. if so, it has to trigger some kind of *planning* to determine what changes to inject into the current behavior of the system such that it returns to the equilibrium state within the new environment, This planning would require *knowledge* to select the right behavior from a large set of possible behaviors to counter the change. Finally, the motor neurons *execute* the selected change.

Even though the authors attempt to make concrete this definition with an example, the example is not particularly helpful and seems rather contrived: dynamic loading and unloading of components in a forest fire simulation. That dynamic component loading is an example of self-configuration in autonomic systems is possible, and that it is was helpful for this implementation is also possible, but far simpler implementations not based on self-configuration, autonomic computing, or dynamic component loading also seem perfectly sufficient here.

As with the previous chapter, my concern here is that, in this exercise of abstraction, it seems like all the useful concrete information has been abstracted out. The biological analogy, strictly speaking, may be true, but it lacks the power of a true foundation to reinforce existing ideas and ease the conceptualization of new ones. Ultimately, it's not clear to me that the ideas described here will help researchers in the distributed systems community build better autonomic computing systems.

Review

Despite containing many intriguing ideas on an amazing variety of topics, this book ultimately fails its readers on many counts: unclear and vague explanations, which make the book very difficult to read; inadequate or incorrect evaluations, which leave many doubts about the technical validity of described approaches; redundant passages, which waste time and confuse; and, so on. Although the book makes many ambitious promises, it disappoints readers on nearly every one.

In addition to the chapter-specific comments above, there are two generally applicable points worth making here:

- A major problem throughout the book is its lack of consistency, which I suspect is exacerbated by its organization as a handbook. This problem manifests itself in a number of interconnected ways: the separation of related chapters to distant parts of the book, the redundancy between related chapters, the lack of consistent notation, and, most importantly, the dramatic variance in quality between contributions. With aggressive re-editing, re-organization into logical sections, and a possible move away from the handbook format, this could be made into a very useful book.

- Another major problem endemic to this book is the lack of good empirical evaluation. Most chapters I read failed even the most basic standards of scientific rigor, and, unfortunately, a few chapters contain particularly egregious examples. Even more worrisome is the publication of what seem like novel results. Instead of being published as a chapter in a book, it seems that such results should instead be published in a journal or conference and undergo proper peer review.

All in all, my feelings toward this book can be best described as disappointed. As a scientist with backgrounds in both biology and machine learning, I deeply appreciate the need for books that bridge the two subjects and especially one that introduces biological ideas to computer science. This book does not meet that need in its current form, but it contains many valuable ideas just waiting to be molded into something more useful.

References

[1] Francisco A. Camargo. Learning algorithms in neural networks. Technical Report CUCS-062-90, Columbia University, 1990.

[2] Erick Cantu-Paz. A survey of paralle genetic algorithms. *Calculateurs Paralleles*, pages 141–171, 1998.

[3] Marco Dorigo and Dianni Di Caro. The ant colony optimization meta-heuristic. In David Corne, Marco Dorigo, and Fred Glover, editors, *New ideas in optimization*. Mcgraw Hill, 1990.

[4] Stephen Olariu and Albert Y. Zomya (editors). *The handbook of bioinspired algorithms and applications*. Chapman and Hall, CRC, 2006.

[5] Wikipedia. Genetic algorithms. `http://en.wikipedia.org/wiki/Genetic_Algorithms`.

[6] Wolfram. Cellular automata. `http://mathworld.wolfram.com/CellularAutomata.html`.

[7] Xin Yao. A review of evolutioaly artificial neural networks. *International Journal of Intelligent Sytems*, pages 539–567, 1993.

Review [18] of
Theoretical and Experimental DNA Computation
by M. Amos
Published in 2005 by Springer-Verlag Berlin Heidelberg, 172 pages

Reviewer: Maulik A. Dave

1 Overview

Interest in DNA computing has increased among research scientists since 1990s. Author himself is a scientist working in DNA computing. The book is an overview of DNA computing. It touches both theoretical and experimental aspects. The theoretical aspects include algorithms, and computing models. The experimental aspects consist of various DNA experiments done for computing.

2 Content Summary

The book is divided into 6 chapters.

2.1 DNA molecule and operations on DNA

The first chapter gives a general introduction to DNA. A brief explanation of the double helix structure of DNA is followed by a detailed explanation of operations on DNA. The terms such as synthesis, denaturing, and ligation are explained. Cloning of DNA is described in details. Techniques such as Gel Electrophoresis for sorting DNA by size, Restriction Enzymes for recognizing specific DNA, and Polymerase Chain Reaction for amount of DNA in a given solution, are described.

2.2 Theoretical Computer Science

The second chapter introduces theoretical computer science. It begins with a brief section on gates, and boolean algebra. Models such as automata, Turing machine, and Random Access machine are described. Algorithms, and data structures are introduced. An introduction to computational complexity theory includes the NP completeness.

[18] © 2009, Maulik Dave

2.3 DNA Computational Models

The models are referred as molecular because they described at abstract level. The models are classified into filtering models, splicing models, constructive models, and membrane models. In filtering models, a computation consists of a sequence of operations on finite multi-sets of strings. With other models, a filtering model made by author, called parallel filtering model, is described in details. The parallel filtering model has remove, union, copy and select operations. NP algorithms written using these operations are also described. Splicing models has a splicing operation, which takes two strings, and concatenates prefix, suffix of one another. Constructive models utilize principal of self assembly. Models, which are chemical, or biological, but not DNA based are described under the term membrane models.

2.4 Complexity of algorithms in DNA Computing

The chapter on complexity presents more insight into the DNA computing. It discusses in details, two boolean circuit models. A DNA computing model, which is able to simulate boolean circuits, is a Turing-complete model. The discussion on such models include introduction to the model, laboratory implementation, an example application, and complexity analysis. Later in the chapter, P-RAM simulation on DNA computing is discussed. Processor instruction set, overview of underlying controller, and translation process from P-RAM algorithms to DNA implementation are described. Further, the P-RAM model is explained by using List ranking problem as an example.

2.5 Laboratory experiments

Laboratory implementations of the algorithms are described with their actual experimental details in last two chapters. Author's own work is described in more details. It is compared with Alderman's work. The chapters do not confine to the DNA computing only. It describes implementation of Chess games on RNA. Groups of organisms having two nuclei, and possessing hair like cilia for movement are called Ciliates. A small chapter is devoted for describing computing based on ciliates.

3 Conclusion, and Comments

A high school level knowledge of biology, particularly of DNA is sufficient to understand the experimental descriptions in the book. However, to appreciate the computing part of DNA computing, knowledge of computers at undergraduate level is necessary. The book is a good introduction to DNA computing for both new researchers, and readers having general interests.

Review of[19]
Coding for Data and Computer Communications
by David Salomon
Springer, 2005
548 pages, Hardcover

Review by
Fatima Talib (fatima.talib@acm.org)

[19]©2009, Fatima Talib

1 Introduction

Coding for Data and Computer Communications combines three major themes: reliability, efficiency, and security in data storage and transmission. The book content is elaborate and diverse such that it cannot be categorized under one specific area of computer science but, as the title suggests, it lies in the intersection of some of the most vivid areas of modern computing and information sciences including information theory, cryptography, networking, and information and communication technology, all of which draw heavily from mathematical disciplines such as combinatorics, statistics and probability, and discrete structures. The book provides a great deal of information in each of the previous aspects of data coding and communication as well as the design tradeoffs inherent in attempting to optimize the speed, security, and reliability in coding and communicating data.

This book is for the general audience of computer scientists and for computer science students and teachers. Researchers and specialists in networking and internet-security will find it informative. Professors who teach various aspects of data coding in their courses may also find it a useful textbook. Readers who are interested in cryptography, data compression and transmission, digital data representation, or simply a new way of looking at information and data using codes and numbers can learn a lot from this book, given that they have an adequate mathematical background.

2 Summary

The book is divided into three main parts paralleling the themes of reliability, efficiency, and security. These are:

- Part I: Channel Coding (error-control codes: error- detecting and error-correcting codes)
- Part II: Source Codes (data compression)
- Part III: Secure Codes (cryptography)

The following sections briefly present the content of each chapter.
Part I on Channel Coding contains the following chapters:

2.1 Chapter 1: Error-Control Codes

This chapter starts by explaining the various processes that data go through from the time it originates from its source until it reaches its destination. The chapter introduces the main concepts of redundancy, parity and check bits, Hamming distance and presents Hamming codes, periodic codes, and Reed-Solomon codes (decoding for the latter is not explained as it is "complex and beyond the scope of this book" but an outline of the decoding process is given). It distinguishes techniques used for error correction from those used for detection only. Three sections focus on compact disc (CD) error control and data coding since this medium is considered error-prone. For more than 1-bit error correction, the chapter explains the elegant concept of a generating polynomial. The chapter concludes with a discussion of some geometric constructions used to generate codes. These are Golay codes and the method of projective planes.

2.2 Chapter 2: Check Digits for Error Detection

While the discussion in chapter 1 is mostly on binary numbers (based on finite fields), this chapter discusses error detection methods used for decimal data. Error detection in many applications using decimal numbers is based on a single check digit appended to the number. This chapter explains how this digit is computed and then used to detect many common errors in applications such as

ISBN, different types of barcodes, and the unique identification numbers found on postal money orders, airline tickets, bank checks, and credit cards. The chapter presents the IBM check digit which is a more elaborate error-detection scheme based on the idea of permutations and used for an arbitrary number of data digits. The chapter also discusses the case of two or more check digits in sections 2.9 and 2.10. Another generalization of the use of check digits is given in 2.10: The Verhoeff Check Digit Method; the section describes an algorithm devised by Dutch mathematician Jacobus Verhoeff in 1969 that detects all single-digit errors and adjacent-digit transpositions in ID numbers of arbitrary length.

Part II on Source Codes discusses two main methods for data compression each presented in a different chapter and concludes with a chapter dedicated to image compression techniques as follows. In the discussion of the topics above, the chapter introduces the concepts of adaptive (dynamic) and non-adoptive compression, lossy vs. lossless compression (some fraction of the original data is lost vs. the original data is perfectly reconstructed in decoding). This part of the book introduces some measures used to indicate the compression performance and uses these measures to evaluate the performance of the codes presented.

2.3 Chapter 3: Statistical Methods

Statistical data compression exploits the statistical properties of the data being compressed; these statistical properties include the frequency and so the probability of each symbol (or string of symbols) in the data. This chapter starts by introducing the concept behind basic statistical methods used in data compression using probabilities and the notion of entropy. Variable-size codes are presented with emphasis on the intuition that compression is achieved using variable-size codes when short codes are assigned to frequently occurring symbols and long codes to the rare symbols. In this chapter, the opposite process of decoding is also explained. Various aspects of Huffman coding as an example of a well-known algorithm for generating a set of variable-size codes with the minimum average length—are discussed in this chapter. Facsimile compression is presented as an application using statistical data compression. Arithmetic coding is the last topic in this chapter. It represents a sophisticated statistical method that uses symbols frequencies to replace the input data with a single number; this method uses probabilities of the symbols in the data to narrow a certain intervals to a point where specifying the interval requires longer numbers that may be used after some manipulation to represent the input data.

2.4 Chapter 4: Dictionary Methods

The idea of dictionary-based methods for data compression is to select strings of symbols and encode each with a special code (a short token) using a dictionary. The dictionary is used to encode and decode the data. The word dictionary should not give the impression that this method is used only in compressing natural language text; the method is used to compress program files and other types of data. Dictionary-based methods are based on analyzing the input data (search and match operations), building the dictionary (the code), encoding, and decoding. This chapter discusses the development and features of a popular dictionary-based method used in many compression software: LZW. The development of this method is rooted in the LZ77 (sliding window) method, and its subsequent variants LZSS and LZ78. The main difference between these methods is in deciding what phrase to add to the dictionary.

2.5 Chapter 5: Image Compression

The chapter begins by stressing the special need for compression in handling image files since they tend to be too large and hence slower to transmit over a communication channel. The chapter discusses the differences between various image types and the compression approach used for each. The chapter is appended with a discussion of JPEG: a sophisticated method used to compress grayscale and color still images. It is worth mentioning that JPEG is considered, at least in this context, as a compression method, whereas the graphics file format that specifies the image features is referred to as JFIF (JPEG file interchange format).

2.6 Part III: Secure Codes

This is the largest part of the book comprising 10 chapters that can be grouped into three main themes:

Introduction to cryptology and a discussion of simple classical ciphers including the Caesar cipher, affine cipher, and the "one-time pad" cipher. The chapter introduces basic principles of secure codes such as Kerckhoffs' principle that states that the security of an encrypted message should depend on the secrecy of the key not on the secrecy of the encryption algorithm.

Each of the five chapters that follow (7 through 11) is dedicated to a specific type of cipher:

- Mono-alphabetic Substitution Ciphers: these are ciphers based on replacing each symbol in the input alphabet with another symbol (of the same or different alphabet) and the replacement rule does not vary
- Transposition Ciphers: simply put, these are ciphers based on replacing the sequence of input symbols with a permutation of itself
- Poly-alphabetic Substitution Ciphers: these are similar to mono-alphabetic substitution ciphers except that the replacement rule changes
- Stream Ciphers: ciphers in which each bit of the input data is encrypted separately
- Block Ciphers: ciphers in which the input data is divided into blocks and each block is encrypted separately by a combination of substitution and transposition. Lucifer, an actual block cipher is examined in this chapter. The chapter closes with a discussion of the development of a popular encryption method: the Data Encryption Standard (DES).

Chapter 12 on public-key cryptography discusses the development of methods for secure key-exchange over non-secure channels; major titles are the Diffie-Hellman-Merkele keys, RSA cryptography, and PGP (for Pretty Good Privacy: the software written to realize the potential of the RSA public-key cipher). The chapter also presents threshold schemes which are based on the idea of maintaining secrecy by dividing the secret message among several persons (components) such that a subset (or subsets with certain size) of them could retrieve the secret by combining the individual parts. Several authentication protocols are also discussed in this chapter. The last two sections of the chapter shed light on Secure Socket Layer (SSL) and Message Digest 5 Hashing (MD5), respectively. The first is a protocol used to secure data sent over the Internet and the latter is a hashing function that serves as a secure and efficient message digest for digital signature applications.

Chapter 13 is on data hiding (steganography). Steganography is another level of securing data by hiding the data itself not only its meaning as in cryptography where the meanining of the data is hidden but the fact that there is encrypted data is not hidden. The chapter introduces data hiding and its basic features and applications (such as watermarking). Chapter 14 presents data hiding techniques used in images and chapter 15 presents other methods of data hiding used for video and audio file. The chapter discusses public-key steganography and the possibility of having a system that provides absolute security in steganography. The last section provides examples of current steganography software.

The book is appended with learning material on Symmetry Groups (Group Theory), Galois Fields, and Cyclic Redundancy Codes (CRC). A following appendix provides a list of research and programming projects for interested readers or for potential use as a practical component of a course on some aspects of data coding.
At the end of book, each of the Glossary, Bibliography, and Index of terms and authors indicates the vast amount of knowledge traced and presented and the effort made to acknowledge the scientists and researchers who contributed to and inspired the field.

3 Opinion

Although the preface describes the book as "an attempt to bring the three disciplines of data compression, error-control codes, and cryptography and data hiding to the attention of the general public," it can be fairly said that it gives a relatively smooth and balanced presentation of these three topics. In my opinion, one of the major contributions of the book is the connection it consistently makes between concepts and topics within each of these themes and the linkage between concepts across the themes, thus drawing a clear picture for the reader on how the different parts fit together. Perhaps the significance of this becomes clear when considering the depth and technically different scope of these topics. Moreover, cryptography is an ancient discipline compared to information theory from which the modern topics of source and channel codes have originated. These topics, therefore, vary not only in technical scope and nature but also in maturity and historical development.

Further, the book gives insight into observations and findings in other fields on which some notions and developments in data coding are built. For example, in chapter 2, where error detection in numeric data is presented, the technical discussion opens with findings of a study on the nature and frequency of common errors made by humans when dealing with decimal numbers. Also, in channel and source coding, many lexical and syntactical features of natural language that are exploited in textual data compression and encryption are discussed. Similarly, a discussion of some features of the human perception of color and shapes are introduced when discussing some techniques of (lossy) compression used for image and video files. In fact, early in the book the general analogy between redundancy in natural language and in data files is made and followed by an explanation of how (human) brain exploitation of redundancy in natural language can inspire computer exploitation of redundancy in data files for compression purposes.

The "coding-theoretic" treatment of information emphasizes the new perspective in which information is viewed as a quantitative concept. This concept has been stated in the book and supported by the technical discussion in which information is quantified.

In about 450 pages, the author covers (in considerable depth) a breadth of topics in the three themes of reliability, efficiency, and security in data storage and transmission. The book is, therefore, a substantial read. In addition to basic knowledge in combinaotrics, number theory, abstract algebra, and statistics and probability—to understand the examples in the book, one requires a background in computer science as some examples employ core CS concepts such as finite state machines and some data structures (e.g., trees). From an instructional design point of view, however, the book can be reasonably used by an independent learner (given adequate background in computer science and mathematics) as well as a basic or reference textbook for a university course. This is due to the progressive development of concepts, the exercises that are in-line with the text (with answers at the end of the book), and the interesting visualization of some of the abstract concepts (e.g, the geometric method of projective planes to generate codes— discussed at the end of chapter 1) and other visual components and illustrations. Examples are given in almost every section of the book. An entire appendix is dedicated to projects which can be proposed as a practical component of a course using this book as a standard or a reference text or as a further research guide for interested readers.

Mentioning research, the book poses interesting questions as it presents and elucidates aspects of data coding and communication. In chapter 2 for example, where Hamming codes are discussed, the author begins the discussion of single-error correction double-error detection code in section 1.9 (page 16) by the following paragraph: "Even though we can estimate the length of a 2-bit error-correcting Hamming code, we don't know how to construct it"?! In the last chapter of the book, the author puts forward the question of whether there exists (in principle) a system in steganography that provides absolute security (in data hiding) just like the one-time pad cipher provides absolute security[20] in cryptography?

To do justice to the thoughtfully structured and presented wealth of information in this book, I would say that in an attempt to communicate the essential features of Salomon's "Coding for Data and Computer Communications," this review is—in the squeezing sense of compression[21]—a very compressed one.

[20]The absolute security of this cipher is subject to some well-defined conditions.

[21]which, contrary to the common thought, not the sense of compression used by compression algorithms/software as explained in the book!