The Book Review Column[1]

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

Welcome to the Book Reviews Column. We hope to bring you at least three reviews of books every month. Three of the books reviewed in this column are from the DIMACS series. There will be more reviews of such books in the future. These books offer snapshops of a subfields of theory. I personally urge you to urge your library to subscribe to these books. It is a great way to keep up with recent results in Theoretical Computer Science.

1. **Mathematical Support for Molecular Biology** , edited by Martin Farach-Colton, Fred S. Roberts, Martin Vingron, and Michael Waterman. Reviewed by: Chrystopher L. Nehaniv. This book is a collection of fourteen articles from the DIMACS special year on Math and Molecular Biology. Of these, five are on implementations of algorithms for biological problems.
2. **DNA Based Computers II**, edited by Lauara F. Landweber and Eric K. Baum. Reviewed by Mitsunori Ogihara and Animesh Ray. This book consists of twenty papers from the second DIMACS workshop on DNA computing.
3. **DNA Based Computers III** edited by Harvey Rubin and David Harlan Wood. Reviewed by Martyn Amos. This book consists of twenty papers from the third DIMACS workshop on DNA computing. This workshop was a year after the second one.
4. **Online Computation and Competitive Analysis** by Allan Borodin and Ran El-Yaniv. Reviewed by Neal Young. Online problems are problems (like memory allocation) where the data is coming at you and you need to decide NOW what to do. The measure of how well you do is to compare it to how well you would do if you knew ahead of time what the data was going to be. This is a textbook/referece book on the topic.

Review of Mathematical Support for Molecular Biology[2]

Series: DIMACS Series in Discrete Mathematics and Theoretical Computer Science (ISSN: 1052-178) Vol. 47
Editors : Martin Farach-Colton, Fred S. Roberts, Martin Vingron, and Michael Waterman
Publisher: American Mathematics Society, 1999
ISBN: 0-8218-0826-5
xv+288 Pages, Hardcover, $65.00
Reviewer: Chrystopher L. Nehaniv (c.l.nehaniv@herts.ac.uk)

Mathematics has grown and developed through its association with the natural sciences, of which, historically, physics has been most influential. Currently new scientific knowledge and new mathematical inspiration is coming from the more recent collaborations between mathematics and the biological sciences. This is exemplified by growing field of computational molecular biology which relies on discrete mathematics and theoretical computer science for its analyses, especially of

[1]© William Gasarch, 1999.
[2]© Chrystopher L. Nehaniv, 1999

the evolution and properties of biological sequences such as those of DNA (and its cousin RNA – which are each comprised of sequences of four possible nucleotides) and proteins (amino acid chains, abstractable for some purposes as sequences over a 20-letter alphabet). This important domain is addressed by the volume under review, but the reader should know that scope of mathematical contact with biology (*bioinformatics* or *mathematical biology*) is much broader including, for instance, mathematical modelling of morphogenesis [5, 4], population genetics, ecology and spatial effects [1, 7], regulatory control within medicine and biological systems [4, 2], mathematical hierarchies in evolution and in the structure of biological systems [3, 5], or areas of engineered sequences (such as DNA Computing, e.g. [8]), among many other areas.

This volume features fourteen papers from the "DIMACS Special Year on Mathematical Support for Molecular Biology". Interestingly this "special year" spans the five years 1994-1998. The NSF Science and Technology Center in Discrete Mathematics and Theoretical Computer Science (DIMACS) hosted it to expose discrete mathematicians and theoretical computer scientists to problems of molecular biology that their methods could usefully address, to make molecular scientists aware of mathematical scientists eager to help them solve their problems, to support concentrated exposure of young researchers to this interdisciplinary area, and to forge long-lasting collaborative links between the biological (especially molecular) and mathematical scientific communities via a series of workshops, vistor programs, and postdoctoral fellowships. This volume is a "snapshot" of this active area of research that collects overview and research papers from the special year, as well as including five papers following the associated DIMACS Algorithm Implementation Challenge, which aimed to bridge the gaps between theory, computer implementation, and practical application.

W. M. Fitch's contribution, "An introduction to molecular biology for mathematicians and computer programmers" provides a valuable introduction to biological concepts for understanding the problems of constructing evolutionary trees from linear biological sequences, including that of *alignment*, i.e. lining up the letters in a sequence with homologous ones– those having a common ancestral nucleotide or amino acid – in another related sequence in order to compare them and measure their evolutionary distance, leading to the construction trees depicting phylogenetic relationships between nodes representing related sequences.

D. Gusfield and L. Wang's paper, "New uses for uniform lifted alignments", addresses algorithmic problems that arise in such efforts to determine molecular evolutionary histories and to align more than two sequences, and in particular to deduce internal node labels representing reconstructed ancestral sequences (*phylogenetic tree alignment*).

"Sequence alignment and phylogeny construction" by M. Vingron reviews ideas linking multiple sequence alignment and phylogeny reconstruction, focusing on commonalities between character-based methods (intensely studied by computer scientists) and hierarchical profile-based methods (prevalent in biology).

"A new look at tree models for multiple sequence alignment" by D. Durand addresses cases arising in biological data where the tree model for multiple sequence alignment fails due to convergent evolution, difficulties with data, and other factors.

"Sequence alignment in molecular biology" by A. Apostolico and R. Giancarlo surveys some criteria widely used in sequence alignment and comparison problems, and some of the corresponding solutions.

D. B. Searls's "Formal language theory and biological macromolecules" reviews the use of grammars, parsers, and automata in such biological applications as recognizing genes and other higher-order features, and describes work on their application to analysis of the structural dependencies arising by virtue of the biochemical properties of nucleic acids or amino acids in biosequences.

The structural theme (molecular modelling leading to the prediction of shapes) is also taken up by C. A. Floudas, J. L. Klepeis, and P. M. Pardalos, who report on "Global optimization approaches in protein folding and peptide docking" that address protein conformations and binding interactions between macromolecules by examining aspects related to conformational energy. The paper includes nearly two hundred references to experimental and theoretical work in this area.

C. J. Benham studies "The topologically driven strand separation transition in DNA—methods of analysis and biological significance" addressing control of the locations and occasions of local separations of the two strands of a DNA double-helix, and methods for predicting such separation on the basis of stress energetics and (nonlocal) topological constraints on the molecule.

C. L. Smith, T. Sano, N. E. Broude, and C. R. Cantor treat "Parallel strategies for DNA manipulation and analysis", reviewing DNA structure and properties, and illustrating some methods for manipulating DNA that allow massively parallel strategies to be used in biological experiments and possible future applications of computing with DNA to problems outside biology.

The remaining five papers arose from the DIMACS Algorithm Implementation Challenge already mentioned and address two application areas for implementations of computational support for molecular biology backed up by rigorous mathematics.

The first two papers address sorting possibly signed permutations by reversals. A. Caprara, G. Lancia, and S.-K. Ng's well-written paper on "A column-generation based branch-and-bound algorithm for sorting by reversals" serves as nice introduction to the topic of sorting by reversals which is applicable to the comparision of genomes, at the level of orderings of genes, rather than the lower level of the structure of particular nucleotide sequences associated with with genes.[3]

E. M. Jordan attacks the problem of "Visualizing measures of genetic distance", especially those arising from sorting permutations, whose associated "alternating graphs" or "dart graphs" are related to "rotation graphs", which correspond to two-dimensional (possibly non-orientable) topological surfaces. He shows that derivation of certain lower bounds for sorting is equivalent to consideration of quantities related to the Euler characteristic of these associated surfaces, namely the number handles and cross-cap number. Moreover, his software allows one to visualize these surfaces.

The final three papers in the volume describe software that attempts the re-construction of very long DNA sequences (*contigs*) of various organisms from numerous short *fragments*. This is the important *fragment assembly* problem which arises from the "shotgun method" for sequencing that involves radiation or chemical treatment that breaks DNA into numerous fragments whose relative positions and reading direction is unknown, and may be complicated by errors such as insertions, deletions, and mismatches. These papers include various insights and performance illustrations on biological data and are L. Milanesi, M. Marsilli, G. Mauri, C. Rolfi, and L. Uboldi's "Fragment assembly system for DNA sequencing projects", X. Huang's "Performance of the CAP2 sequence assembly program" and J. Meidanis' "A simple toolkit for DNA fragment assembly".

The present volume offers an excellent view of the kinds of work in theoretical computer science and discrete mathematics that are particularly applicable to the explosively growing area of molecular biology and genomics, and the overviews provide useful surveys explicating key concepts and techniques and will serve as inroads for those aspiring to enter research by providing a guide to the voluminous literature in the field. However, one should not expect the field to stand still: New problems as well as new solutions to old ones arise daily. So this is a book to be read while

[3]It seems to the reviewer that the important evolutionary phenomena of duplication and deletion of entire genes would warrant generalization of the permutation approach to the considerations of mappings of a genome in which the number of genes might also be allowed to increase or decrease. In particular, semigroup-theoretic methods extending the current group-theoretic methods are called for.

it's still hot! The book will be of interest to both experts and novices – especially theoretical computer scientists and computational molecular biologists – seeking information on the state of research. Readers wanting to go further into broadly grounded mathematical biology would do well to consult also J. D. Murray's general biomathematics text [4], and for further details on molecular computational biology – the book of M. S. Waterman [9], one of the editors of the reviewed volume, as well as the R. D. M. Page and E. C. Holmes text on molecular evolution [6].

References

[1] Robert H. Gardner (Ed.), *Predicting Spatial Effects in Ecological Systems*, (Series on Lectures on Mathematics in the Life Sciences, Vol. 23), American Mathematical Society, 1993.

[2] F. C. Hoppensteadt and C. S. Peskin, *Mathematics in Medicine and the Life Sciences*, (Texts in Applied Mathematics, Vol. 10), Springer Verlag, 1992.

[3] Boris Mirkin, F. R. McMorris, Fred S. Roberts, and Andrey Rzhetsky (Eds.), *Mathematical Hierarchies and Biology*, (Series on Discrete Mathematical and Theoretical Computer Science, Vol. 37), 1999.

[4] James D. Murray, *Mathematical Biology*, (Biomathematics Texts, Vol. 19), 2nd, corrected edition, Springer Verlag, 1993.

[5] Chrystopher L. Nehaniv (Ed.), *Mathematical and Computational Biology: Computational Morphogenesis, Hierarchical Complexity, and Digital Evolution*, (Series on Lectures on Mathematics in the Life Sciences, Vol. 26), American Mathematical Society, 1999.

[6] Roderic D. M. Page and Edward C. Holmes, *Molecular Evolution: A Phylogenetic Approach*, Blackwell, 1998.

[7] Jonathan Roughgarden, *Theory of Populations Genetics and Evolutionary Ecology: An Introduction*, Prentice Hall, 1996.

[8] Harvey Rubin and David Harlan Wood (Eds.), *DNA Based Computers III*, (Series on Discrete Mathematical and Theoretical Computer Science, Vol. 48), 1999.

[9] Michael S. Waterman, *Introduction to Computational Biology: Maps, Sequences, and Genomes*, Chapman & Hall, 1995.

Review of
DNA Based Computers II[4]

Series: DIMACS Series in Discrte Math and TCS Vol 44
Editors : Lauara F. Landweber and EricK. Baum
Publisher: American Mathematics Society 1998
ISBN: 0-8218-0756-0
275 Pages, Hardcover, $59.00

Reviewers: Mitsunori Ogihara (ogiharacs.rochester.edu) and
Animesh Ray (rayar.biology.rochester.edu

DNA based computation is a four year old paradigm of parallel computation that is yet to see a real world application. It is difficult to predict the future direction of the field. While researchers working in this field are well aware of the current development, there is an urgent need for infusion of fresh ideas. Unfortunately there is no adequately accessible book that covers the entire spectrum of interest, from computer science to molecular biology, necessary for kindling the interest of an

[4]© Mitsunori Ogihara and Animesh Ray, 1999

uninitiate. The present book does not fulfill this need. It does, however, provide a compendium of ingenious articles representing the knowledge of this field as it was in 1996. As such this book, though mainly of a historical interest, may provide an informed student with a flavor of theory and experiments in this field. Handsomely bound, the book has twenty original contributions from researchers attending the DIMACS Workshop in June 1996. The editors, a molecular evolutionist and a computer scientist, have done a good job of putting the papers together. One wishes that the volume came out a little earlier.

Three basic topics are covered by the contributions: design of molecular algorithms (ten articles), design of computation models with molecules (ten articles), and the technical issues in dealing with molecules in this context (ten articles). Obviously some articles overlap these artificial distinctions.

The first paper (pages 1–29), a contribution of Len Adleman's group, addresses the question: is it possible to devise a computing machine with DNA using only sequence-specific hybridization as the central processing paradigm? The proposal (the sticker based model) begins with the generation in one step of all possible bit strings of DNA by an elegantly simple method. Here a long single strand of DNA (length N bases) is arbitrarily partitioned into K non-overlapping segments each M bases long. The long strand is the memory strand. Corresponding to each of K segments, each of which is a unique bit position, there is an M base long complementary DNA sequence called sticker. If a sticker is hybridized to the corresponding bit position on the memory strand then the bit is ON, otherwise it is OFF. A number of operations on the bits are defined: combine (two sets of bit strings are merged into the same test tube), separate (a set of strings sharing the same bit value corresponding one or more bit positions are separated from the remainder), set a bit (attach a sticker to all strings in a collection corresponding to a bit position), and clear a bit (remove the sticker corresponding to a particular bit position from all memory strands). All of these operations are practically reasonable except the last one. The authors do suggest a potential method for performing clear operation using stickers made of peptide nucleic acids (PNA) instead of DNA, but experimental validation of this proposal is yet to come. Without the bit clear operation manipulation of bits is one-way so direct simulation of silicon-based computers is not possible. However, the library generation, which is used in parallel brute force search in solving NP problems using DNA, is dramatically simplified. The article describes in detail how these elementary operations on bits may be carried out sequentially, but in parallel to all memory strands, to perform computation of the NP-complete Minimal Set Cover Problem. For the first time, this paper examines the issue of how to minimize the operational errors inherent in molecular biological operations from an engineering perspective. While it is unlikely that DNA computers exclusively based on stickers will ever be a reality, this proposal is a significant contribution to our thinking of the issues involved. Finally, the method of producing all 2^K bit memory strands in parallel and in one step may find applications in other fields such as combinatorial chemistry.

Initial computations with DNA strands assumed only linear self assembly as a mechanism of information exchange. DNA, however, can self assemble into two- and three-dimensional branched structures. Do these higher order self-assemblies provide a computational advantage? This question was addressed in the contribution by Winfree and his collaborators (pages 191–214). They have shown that linear assembly of DNA sequences can generate regular languages, the assembly of branched DNA molecules can generate context-free languages, and, the three dimensional self-assembly of DNA (using branched structures growing in three dimensions) can produce universal languages (recursively enumerable languages). The proof is easily extendible to all informational polymers. The three-dimensional self-assembly of many biological molecules can generate recursively enumerable languages. Furthermore, homologous and site-specific recombination (including all forms of transposition), the transcription of RNA on DNA, the translation of polypeptides on RNA, and RNA splicing (discussed separately by Landweber, pages 181–189), all generate branched (not necessarily covalent) intermediates of polymers; therefore all can potentially generate context-free languages. A companion paper by Seeman *et al.* (pages 215–233) illustrates the range of bewildering topologies that can be designed artificially by carefully manipulating DNA strands. This article also provides cautionary notes on the traps and slippery paths waiting for the uninitiate in the practice of tying knots with DNA. The beauty of some of the geometries possible with DNA is striking. One wonders why nature has not made these structures; perhaps this is an indication of how slippery the route to computing with three-dimensional self-assemblies of DNA might indeed be.

DNA based computation appears to be ideally suited for some kind of search problems. Instead of brute-force search as had been proposed earlier, Baum and Boneh (pages 77–85) showed how algorithms with dynamic programming can be implemented on DNA. Suppose that the essential part of computation by dynamic programming is building of a multi-dimensional Boolean table and that each object in a given instance relates an entry of the table with another such that if one entry is true then the other is false. Then with a method similar to that of Adleman's 1994 proposal (*Science* 266:1021–1024) one can generate all entries of the table that are true in a single biochemical operational step. In order to find the exact solution to the problem one has only to search for the optimal solution. They illustrate this proposal by a DNA-based method for the Knapsack problem. Further exploration of this approach will be important. As another typical problem they take is the Reachability, which computes whether there exists a path between two arbitrary vertices within a directed graph. The method uses DNA based breadth-first search within small subgraphs in parallel, then uses the functions computed from these subgraphs to iteratively search larger and larger graphs. The authors point out a technical problem in this method that if there are few vertices with a small number of paths leading to them compared with most other vertices (an "unbalanced graph"), then these minority paths will be missed due to competition among paths leading to most frequently connected vertices. They provide a theoretical limit to the "balance" of the graph that allows a successful solution. Additionally, a fundamental problem in DNA computation, the problem of "communication" between two minority molecules by random collision, has been raised and examined from theoretical viewpoint.

DNA computation codes information into sequences of DNA. Most of the biochemical operations that are used for DNA computation are based on matching of DNA (DNA-DNA hybridization), which is relatively error-prone. One uses a set of predetermined DNA sequences of the same length as the basis for coding and concatenates them to generate longer patterns. It is crucial that the base collection has the property that no two elements from it are close to each other in terms of encoding. How can one select the basis? One suggestion for the "unique" encoding is to let each pair disagree with each other at a certain number of positions at least. The paper by Baum (pages 235–241) presents a combinatorial formula in two parameters n and k that provides the number of length n patterns that each pair disagree with each other by at least k positions.a The paper by Deaton *et al.* (pages 247–258) uses a more complicated condition for two strands to be far apart. They suggest the use of genetic programming for searching the base set and show some experimental results.

The article by Lipton (pages 259–265) makes a few important suggestions. First DNA computation resembles that of pebble games. Second the use of both destructive and nondestructive detect operations will enrich DNA computation model. Third DNA computation can naturally implement the concepts of global memory and local memory in parallel computation.

Most DNA computational models assume that interacting molecules are able to diffuse in three dimensions. This, coupled with redundancy of memory molecules, ultimately gives the power of massive parallelism. A drastically different way to view molecular computation is by allowing interactions only on two dimensions. While the effective physical search space is reduced, surface based computation has two advantages: it is possible to use encoding based on spatial coordinates, and, second, many molecular reaction rates are enhanced due to steric effects when they are constrained to take place on surfaces. The contribution by Liu *et al.* (pages 123–132) reports the results of actual preliminary experiments with operations on surface-immobilized DNA strands.

It is clear that more new, innovative, ideas will infuse into this field, perhaps as later additions to this volume. All molecular computations so far have this in common: massively parallel search is made in chemical solutions using the property of information transfer by biological molecules. As the editors wistfully observe, molecular computation provides a kind of “solutions of solutions”.

Review of DNA Based Computers III[5]

Series: DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Volume 48
Editors: Harvey Rubin and David Harlan Wood
Publisher: American Mathematical Society , 1999
ISBN: 0-8218-0842-7
375 pages, Hardcover, $79
Reviewer: Martyn Amos (martyn@csc.liv.ac.uk)

The concept of computing at a molecular level back to Feynman in the 1960s, and was developed by Conrad and others in the 1980s. Recently, there has been a surge of interest in the field as a result of Adleman’s 1994 paper, which showed how molecular level computation may be achieved using standard laboratory operations on strands of DNA.

Since Adleman’s paper, DNA computing has developed into a rapidly growing area of enquiry; several major conferences now dedicate special sessions to the field, and the Annual Workshop on DNA Computing held its fifth meeting this year, at MIT. This volume, DNA Based Computers III, contains papers presented at the earlier Third Annual Workshop, held in June 1997 at the University of Pennslyvania. The twenty-five contributions in this volume were collected by the editors, Harvey Rubin, a medical doctor at the University of Pennslyvania, and David Harlan Wood, a Computer Scientist at the University of Delaware. The diverse backgrounds of the editors are a testament to the truly inter-disciplinary nature of the field.

Several “threads” dominating the workshop are represented by various papers in this volume, although this fact is not at all clear on initial inspection of the contents page. However, the preface clarifies matters somewhat, and the different sessions are clearly delineated.

The *Second* Annual Workshop was characterised by a general feeling that the field is an exciting one to be in, but that a lot more experimental work needs to be done in order to establish DNA Computing as a viable discipline in the long-term. Although we are still some way away from this position, the volume under consideration describes encouraging movement towards it. Over one-third of the papers describe the results of (sometimes preliminary) laboratory experiments, and a casual flick through the book reveals a healthy number of gel visualisations! The “Making It Work in the Lab” sessions addressed the “real-world” problems inherent to experiments in DNA computing. Often, these problems are non-obvious, due to the unusual nature of the protocols used or the level of accuracy required. The paper by Wetmur (p. 1-24) provides a useful review of the nature of

[5]© Martyn Amos 1999

DNA hybridization, and describes various approaches to and advantages of limiting experimental conditions in order to minimize the occurrence of "undesirable" molecular complexes. The paper by Hartemink and Gifford (p. 25-37) also focuses on DNA hybridization, this time presenting a a simulation system for optimizing the design of DNA strands used as computational components. As the authors rightly state, "...it becomes clear that many of the proposed computational systems will need to move 'from the blackboard into the laboratory' in order to iron out the subtle details associated with their implementation", and this system may well prove to be an invaluable tool in achieving this. The MIT group of Khodor and Gifford then reports a series of experimental observations on the efficiency of DNA extraction operations (p. 39-46), a central task in any DNA computation. They conclude that the community should adopt a standard set of benchmarks for key primitive operations, such as separation. On a similar note, Chen and Harlan Wood propose in their paper (p. 47-56) a new separation technique with a low error-rate. Their method uses circular DNA strands, rather than the more common linear variant, and the authors claim the main benefits of their approach to be simplicity and improved performance over traditional, affinity-based methods. As the use of circular DNA strands has increased since the 1997 meeting, this method should certainly be reviewed again by practitioners in the field. The paper by Hagiya *et al.* (p. 57-72) then describes the tantalising prospect of "single-pot" evaluation of Boolean formulae. As the number of manipulations of the contents of a tube increases, so does the potential for error, so massively-parallel computations performed without human intervention may well be one characteristic of the elusive "killer application" of DNA computing.

The "Proposed Applications" session begins with the description of a proposed "killer app", suggested by Lipton, Landweber and Rabin (p. 161-172). The idea involves operating on "unknown" sequences of DNA, a technique which may have applications in DNA sequencing and fingerprinting. Although the "DNA^{2}DNA" method may prove beneficial in these areas, its relevance to general computational issues is questionable. The paper by Ellington *et al.* (p. 173-184) then makes the extremely important point that nucleic acids can act not only as passive information storage media, but as functional computational components. Ellington *et al.* describe the exciting possibility of nucleic acids such as ribozymes acting as transistors. This "non-passive" view of nucleic acids is currently being championed by several prominent researchers in the field, and this paper is essential reading for anyone working in or considering research in DNA computing.

The highlight of the "Proposed Algorithmic Approaches" session is a paper by Fraenkel (p. 101-122), in which he discusses the theoretical relationship between the protein folding problem (an incredibly significant problem in molecular biology) and NP-complete problems. He relates the problem of finding the final conformation of a protein (given only its amino acid sequence) to that of finding the ground state of spin glasses. This work implies that it may be possible to synthesise proteins to solve specific instances of the spin glass problem.

The "Models of DNA Computers" session opens with a description of "Local Parallel Biomolecular Computation" by John Reif (p. 217-254). Containing an extensive and extremely useful bibliography, this paper advocates a move away from "distributed parallelism" in DNA computation, where operations are performed in parallel on large numbers of different molecules, towards "local parallelism", where operations are executed in parallel on *each* given molecule. Reif describes methods that draw on developments in nano-fabrication as one possible route towards large-scale assembly of molecular complexes to solve specific problems. Ogihara and Ray (p. 255-264) then describe one possible method for reducing the amount of DNA required to solve NP-complete problems, using a technique known as DNA *counting*. Although interesting, this paper appears rather dated in the light of recent compelling arguments that the main benefit of using DNA computation is unlikely to derive from the solution of NP-complete problems. The paper by Blumberg (p. 265-

279) then addresses the important issue of communication in parallel computation, with particular reference to DNA computation, and presents a direct DNA simulation of the Connection Machine. Conrad and Zauner (p. 281-287) then describe a model of a DNA conformational processor, which utilizes the ability of DNA to switch from one form to another, based on the presence or absence of specific chemicals. These molecules could, potentially, be evolved to perform certain pattern recognition functions.

The final "Computability Using DNA" session, as its title suggests, relates "traditional" computability theory to the area of molecular computing. The paper by Freund, Paun, Rozenberg and Salomaa (p. 297-327) describes the notion of "Watson-Crick Complementarity", which guarantees universal computation in any model of DNA computation that provides sufficient input/output capabilities. Kari *et al.* (p. 329-346) then present several characterizations of recursively enumerable languages using insertion/deletion systems. Insertion and deletion of "words" of DNA is a very common biological phenomenon, and these operations can be used as the sole primitives in DNA computations.

As the field of DNA computing is moving so rapidly, some of these papers may have lost some of their immediate relevance. Although hardly constituting part of the DNA computing "fossil record", this book will be invaluable to researchers in the field, as it contains the various stages of evolution of current ideas. However, it may well prove to be a daunting read to the new student of DNA computing, and the fields equivalent of Goldberg's "Genetic Algorithms in Search, Optimization and Machine Learning" still remains to be written. To conclude, it is perhaps worth noting that the Sixth Annual Workshop on DNA Based Computers (previously always held in the United States) will be hosted by Leiden University, The Netherlands, in the year 2000.

Review of[6]
Online Computation and Competitive Analysis
Authors: Allan Borodin and Ran El-Yaniv
Publisher: Cambridge University Press
Hardcover: ISBN 0-521-56392-5
414 pages

Reviewer: Neal E. Young

1 Overview

Since roughly 1985, theoretical research in the competitive analysis of online algorithms (a subfield of theoretical computer science) has grown tremendously. This book by Borodin and El-Yaniv is the first comprehensive guide to this body of research. The book covers some of the fundamental problems that have been considered during this period: list accessing, paging, the k-server problem, metrical task systems, as well as some more application-oriented topics: in scheduling (bin packing, virtual circuit routing, load balancing) and in investment (portfolio selection). The authors use these problems to summarize the important results in the field (including proofs), as well as to introduce general methods, including the use of potential functions, randomized algorithms (very important in competitive analysis!), and game theory. The final chapter considers competitive analysis in the general context of decision making under uncertainty and formally compares competitive analysis to other theoretical approaches.

[6]© Neal E. Young, 1999

Each of the 15 chapters contains a few (typically 5-10) exercises, usually having the reader expand on, or prove some detail of, the analysis presented in the text. The end of each chapter also presents historical notes and several interesting open problems. The book has an extensive bibliography and index, and presents some, but not many, empirical results.

2 Summary of Contents

Chapter 1 explains what an on-line problem is and defines the competitive ratio of an on-line algorithm — the basic measure of algorithm performance in competitive analysis, defined as the worst-case ratio (over all inputs) of the cost incurred by the algorithm on an input to the cost incurred by the optimal solution for that input.

The list-accessing problem is the leading example. The problem is to maintain a dynamically ordered list of items in response to requests for items, where the cost of a request is the distance the requested item is from the front of the list; at each request, the requested item can be moved forward in the list, and at a cost of one per swap, other items can be rearranged. Move-To-Front is shown to be 2-competitive; the upper bound is given both by the potential function method and also by the list-factoring method; the lower bound is shown by an adversary argument.

Chapter 2 expands on Chapter 1, presenting and analyzing 3 randomized on-line strategies, and general lower bounds, for the list-accessing problem.

Chapter 3 introduces the paging problem — in response to a sequence of requests for items, one must maintain a fixed-size cache k of the items. The cost of each request is 1 if the requested item is not in the cache and 0 otherwise. Six deterministic algorithms are analyzed — 4 are shown k-competitive, 2 are shown not to be competitive at all. The ratio of k is shown to be the best possible for any online deterministic strategy. Some experimental results are presented.

Chapter 4 expands on Chapter 3, presenting and analyzing 2 of the easier-to-analyze randomized online paging algorithms (RANDOM and MARK; the optimally competitive algorithms PARTITION and EQUITABLE are not presented in detail). A general lower bound for randomized online algorithms is also presented.

Chapter 5 presents some of the variants of standard competitive analysis for paging, focusing mainly on the access-graph model, but also stating some basic results in the Markov-chain and diffuse-adversary models. These variants are motivated by the observation that in the standard model, the theoretical competitive ratios of (even optimal) online strategies, especially deterministic ones, are much higher than is typically observed in practice. Some preliminary empirical results are presented.

Chapter 6 summarizes background material in basic game theory: extensive and strategic representations of 2-person, zero-sum games; games of complete vs. incomplete information; various kinds of randomized strategies (mixed, behavioral, and general, linear games and games of perfect recall). The chapter presents one application of game theory, to *memoryless* randomized paging algorithms (memoryless means that each decision of which depends only on the current request and cache contents).

Chapter 7 presents *request-answer* games, a general class of online problems including paging and list-accessing. For randomized strategies, there are at least three reasonable ways to define the competitive ratio — (1) using an *oblivious adversary*, who knows what the on-line algorithm is but not the specific random choices it makes as it proceeds (this is the most commonly used model, and the easiest for the on-line algorithm), (2) using an *adaptive-offline* adversary, who knows the random choices made by the algorithm up to the current time and can use this information as it builds the request sequence (this is the hardest for the online algorithm), or (3) using an *adaptive-*

online adversary, a limited form of the adaptive-offline adversary that must itself decide how to service each request as it is generated (not knowing the future random choices of the algorithm). Chapter 7 presents the following results: for any request-answer game, if there are online algorithms that are, respectively, a- and b-competitive in models (1) and (2), then there is an online algorithm that is ab-competitive in model (3). Furthermore, randomization does not help the online algorithm in model (3).

Chapter 8 continues the study of game theory by focusing on von Neumann and Morgenstern's minimax theorem for 2-person zero-sum games. The authors discuss generalizations of the theorem to infinite games, and how the theorem implies a method (often called Yao's principle) for obtaining lower bounds on the complexity (in this case the competitive ratio) of randomized strategies. A single application (obtaining a lower bound for randomized paging algorithms) is presented.

In **Chapter 9**, metrical task systems (MTS) are considered. This is another general abstraction for many on-line problems, generalizing paging but only some versions of the list-accessing problem. An $O(N)$-competitive deterministic algorithm is analyzed, where N is the number of states of the MTS. Note that for paging, $N = \binom{n}{k}$, where n is the number of distinct items and k is the size of the cache. A more complicated but optimally competitive work-function based algorithm is also presented. An $O(\log N)$-competitive randomized algorithm for uniform MTS is presented, followed by a relatively complicated but seminal poly-logarithmically-competitive randomized algorithm for arbitrary MTS's. Both analyses are given in the chapter.

The infamous *k-server problem* is the subject of **Chapter 10**. The k-server conjecture is that the optimal competitive ratio for any deterministic on-line algorithm for the k-server problem is k. A lower bound of k was known in 1985, but no reasonable upper bound was shown for any on-line algorithm until a decade later.

The problem is to dynamically locate k servers in a metric space in response to requests for service occurring at locations in the space. At each request, some server must be moved to the location of the request, at a cost equal to the distance from the previous location of the server to the requested location. Paging is the special case when the pairwise distances are each 1.

The k-competitive *tree* algorithm and its elegant analysis are presented for the special case when the metric space corresponds to a tree with weighted edges. Two other special cases are presented. Finally, the authors give the relatively involved analysis of the work-function algorithm (the result that essentially settled the problem) showing that it is $(2k-1)$-competitive for the general problem.

Chapter 11 discusses and analyses randomized algorithms for special cases of the k-server problem: when the metric space is the circle; and when the metric space is *resistive*, in which case the HARMONIC algorithm (move a server with probability inversely proportional to its distance to the request) is shown to be k-competitive.

Chapter 12 concerns load-balancing/packing type problems, including: scheduling on restricted, related, and unrelated machines; virtual circuit routing in graphs (a sort of on-line multicommodity flow or call-scheduling problem) and variants; bin packing (4 pp., covering 2 deterministic on-line strategies).

Chapter 13 continues in the vein of Chapter 12, presenting algorithms for call admission and circuit routing, on-line disjoint paths on specific networks, and optical network routing (an extension of disjoint paths). Some empirical results are presented.

Portfolio selection (making money!) is the topic of **Chapter 14**. The first problem considered is *one-way trading*, in which a fixed amount of one currency must be converted to another currency over a period of time in which the exchange rate varies in an interval $[m, M]$. (Conversion back to the original currency is not allowed.) Randomization is shown not to help, and the optimal competitive ratio is shown to be $O(\log(M/m))$.

The second problem considered is *portfolio selection*, in which an initial amount of cash (not cache!) is used to buy and sell securities whose prices rise and fall over a specified time period. Two offline algorithms are considered (buy and hold one stock, rebalance to keep a relative total values of each type of stock constant) and compared on empirical data. Various online algorithms that achieve competitive ratios of $n^{O(m)}$ with respect to the optimal offline strategy *based on constant rebalancing* are described and analyzed (these analyses appear fairly technical).

The third problem considered is two-way trading — the special case of portfolio selection when there are just two securities available. Competitive ratios (versus the optimal offline) in this model are shown to be very bad — exponential in n (the number of trading periods). Restrictions on the adversary are considered, and some empirical results are presented.

Chapter 15 concludes the book with a discussion of competitive analysis in the general context of decision making under uncertainty. The general question is how, for a given problem, to order the online algorithms in order of preference (independent of particular inputs). Competitive analysis is one way to do this, but the general problem has been considered extensively in the past and many other criteria have been studied. Quoting from the book: *"A decision making approach very similar to competitive analysis was known almost 50 years ago. However, for the most part is was superseded by the 'Bayesian approach,' which evolved to be the main tool for most online applications in economics and finance, statistics, and operations research."*

The chapter continues with an axiomatic approach to various decision-making criteria (by a "criteria" we mean a way of ordering the online algorithms by preference; competitive analysis is such a criterion). For example, one might naturally restrict oneself to criteria that do not distinguish between algorithms that are essentially isomorphic in various natural ways. Going further, if one algorithm performs better on every input than another algorithm, surely any reasonable criterion should prefer the former algorithm. By building up natural properties like these, one can eventually pin down a specific criterion. A natural question is which sets of properties lead to which criteria. This is presumably not just of academic interest, because in any given application one might suspect that certain properties are more important than others. After developing this general approach, the authors give some specific examples of criteria and how they apply to a simple on-line problem. The chapter ends with a discussion of utility theory.

3 Opinion

The book covers the last 15 years of research in competitive analysis. I would describe the book as somewhere between a reference book and a textbook for this material, and slightly more of a reference book than a textbook. The book is not completely exhaustive, but most of the core topics and important results are covered. The authors give proofs for almost all of the results they present, but they have taken care to rewrite the proofs so that they are accessible and mutually coherent. Sometimes they give a simpler proof of a slightly weaker result instead of presenting the strongest known result.

I would say a typical researcher in on-line algorithms is likely, but not guaranteed, to find the background material s/he might want, and most computer scientists will find most of the analyses relatively accessible. On the other hand, an instructor using the book as a textbook would have to take some care to select appropriate readings. Most of the book is probably too advanced for advanced undergraduates, but I think there is a subset of the book that covers many of the core topics in depth and that is at a level appropriate for undergraduates. Even for a graduate course the instructor will have to carefully choose the readings. It may be that the students who will benefit most from taking such a course are those interested in doing research in the field, as opposed to those in other fields who are interested in applying the techniques from the field in their own areas.

Most of the exercises in the book are of the form "prove some particular claim from an analysis in the book" or "adapt such an analysis to a different setting" and so on. Additional exercises might be useful.

The book provides a good reference for the ongoing debate about the motivations for competitive analysis. In the introduction the authors outline their view: that (paraphrasing) (a) competitive analysis is an interesting framework for mathematical analysis of algorithms in its own right, (b) it is too soon to expect it to be uniformly worthwhile over all application areas and (c) it is already having practical relevance in some areas such as communication network routing. In Chapter 7 they state, rather cautiously, (d) "if online algorithm analysis is to be more than an ad hoc collection of specific online problems, a more general framework is needed."

The authors have done an excellent job of summarizing the current state of the field. In doing so, I think that they have implicitly stated an agenda for researchers in the field: to develop theoretical models that increase our practical understanding and effectiveness.

4 Some Statistics

chapter	topic	pages	exercises	open questions	pp. historical discussion
1	list access - intro.	23	8	5	3
2	list access - rand.	7	6	3	1
3	paging - det.	11	10	4	1
4	paging - rand.	9	6	2	1
5	paging - variants	23	9	4	1
6	game theory I	19	6	6	1
7	req.-answer games	10	6	5	
8	game theory II	13	2		
9	metrical task sys.	26	5	6	2
10	k-server - det.	31	15	6	3
11	k-server - rand.	18	4	8	1
12	load balance	24	10	6	2
13	routing	37	16	6	3
14	financial	45	26	8	3
15	other criteria	42	10	5	3
total		354	139	74	25