

---

## Preface

### *What is this book good for?*

Imagine you are a computer scientist working in the bioinformatics area. Probably you will be a member of a highly interdisciplinary team consisting of biologists, chemists, mathematicians, computer scientists ranging from programmers to algorithm engineers, and eventually people from various further fields. A major problem within such interdisciplinary teams is always to find some common language, and, for each member of some discipline, to have profound knowledge of what are the notions, basic concepts and goals of the other participating disciplines, as well as of what they can contribute to the solution of ones own problems. This does, of course, not mean that a computer scientist should do the job of the biologist. Nevertheless, a computer scientist should be able to understand what a biologist deals with. On the other hand, the biologist should not do the computer scientists job, but should know what computer science and algorithm engineering might contribute to the solution of her/his problems, and also how problems should be stated in order for the computer scientist to understand them.

This book primarily aims to show the potential that algorithm engineering offers for the solution of core bioinformatics problems. In this sense, it is oriented both towards biologists indicating them what computer science might contribute to the solution of application problems, as well as to mathematicians and algorithm designers teaching them a couple of fundamental paradigms for the analysis of the complexity of problems and, following this analysis, for the design of optimal solution algorithms. Thus the goal of the book is neither to present a more or less complete survey of bioinformatics themes, nor to do programming or introduce the usage of bioinformatics tools or bioinformatics databases. It is a book on *fundamental algorithm design principles* that are over and over again applied in bioinformatics, with a clear formal presentation what an algorithm engineer should know about each of these principles (omitting lots of things that are more intended for a theoretically complete treatment of themes), and lots of case studies illustrating these

principles. The selection of case studies covers many of the milestones in the development of bioinformatics, such as genome sequencing, string alignment, suffix tree usage, and many more, but also some more special themes that are usually omitted in textbooks as “being too complicated”<sup>1</sup>, for example, the quite complicated derivation of the formula computing the directed reversal distance between two signed genomes, as well as a few themes that recently attracted attention, for example, analysis of molecular networks or prediction of RNA pseudoknots.

### ***Book Overview***

Chapter 1 starts with an informal presentation of *core bioinformatics problems*. From this, a computer scientist or mathematician can draw some first impression of what sort of problems bioinformatics deals with. Most of the presented problems are milestones of the development of bioinformatics, and a few are highly relevant problems that have attracted attention over the past few years.

Chapter 2 sheds some light on what sort of *formal algorithmic problems* evolve from the biological problems presented in Chap. 1. Here, a biologist may learn about the basic notions and concepts a computer scientist uses. For some of the presented algorithmic problems, solutions are given already in this chapter or at least some preparations for their solution in later chapters.

Chapter 3 presents one of the most useful algorithm design techniques in bioinformatics (but also in many other applications areas), *dynamic programming* which is based on the fundamental divide-and-conquer paradigm of recursive problem solving. Although being conceptually rather simple and easy to use there are nevertheless several things to be stated about this approach. First, a dynamic programming solution, though being recursively described, is not implemented recursively on a computer, but in an iterative manner, computation proceeding from solutions for smaller instances to solutions of larger instances. Only by this there is a chance to avoid the combinatorial explosion that would be the consequence of a direct top-down recursive implementation. What one has to deal with when applying dynamic programming is the question of how larger problem instances may be reduced to smaller problem instances, and of how solutions of these smaller problem instances may be assembled into a solution of the larger instance. This is most often rather canonical, but can sometimes be more sophisticated. There are a couple of further “tricks” a user should know, as for example building-in of constraints into the functions to be optimized, or avoidance of multiple evaluation of identical terms within function calls by computing and storing such terms in advance (before evaluation of recursive subterms starts) or at parallel to the main recursion.

Having seen dynamic programming as a design paradigm that leads to efficient and exact algorithms in a highly uniform manner (“cooking recipe”),

---

<sup>1</sup> for the reader...?

Chap. 4 contains much more sophisticated approaches. Here, we highlight the role that intelligent data structures and particularly well-chosen representation and visualization of problems play; a phenomenon that is widely known from, for example, mathematics, algorithm design, and artificial intelligence. We illustrate the enormous impact of proper representations on problem solution by four examples: the quite elegant data structure of PQ-trees that almost trivializes the CONSECUTIVE ONES problem occurring in DNA mapping; the powerful data structure of suffix trees with so many applications in bioinformatics that one is tempted to state “no suffix trees, no bioinformatics”; the astonishing way to make least common ancestor queries in arbitrarily large trees cost almost nothing by a clever single preprocessing of the considered tree; the usage of reality-desire diagrams as a visualization tool making the quite long, technical, and hard to read proof of the beautiful Hannenhalli/Pevzner formula for the reversal distance between signed permutations transparent and quite straight-forward.

Having thus presented more involved approaches in Chap. 4 that are by no means as uniform as the “cooking recipe” of dynamic programming from Chap. 3, this should remind us that we have to be aware that bioinformatics problems may be quite hard, as it is usually the case for most scientific fields dealing with algorithmic problems. Chapter 5 now turns to these hard to solve problems. The formal concept expressing hardness is the concept of *NP-completeness* that was independently introduced by Cook [22] and Karp [38] in 1971 who proved a first problem, satisfiability problem 3SAT for a restricted class of Boolean formulas, to be NP-hard. Showing that further problems are NP-hard is done by polynomially reducing a known NP-hard problem to the problems under consideration. Thus, in principle we obtain longer and longer reduction paths from initial problem 3SAT to further NP-hard problems. In the presentation of NP-hard problems in the literature there are often considerable gaps in reduction paths that a reader must accept. We completely avoid any such gaps by presenting for each NP-hard problem a complete reduction path from 3SAT to the problem under consideration. Chap. 5 also presents several NP-hardness proofs that are quite sophisticated, and thus seldom treated in textbooks but only cited from original papers.

Having learned now that we must be aware of an overwhelming number of NP-hard problems in bioinformatics that, as complexity theory tells us, cannot be efficiently solved by exact algorithms (unless  $P = NP$ ), the question arises which sort of relaxed solution concepts could lead to efficient and practically useful algorithms also for the case of NP-complete problems. Here the idea of *approximation algorithms* is central. Such algorithms deliver solutions that deviate from an optimal one up to a certain constant degree, for example, they may guarantee that a computed solution is at most 10% more costly than a cheapest solution would be. Approximation algorithms are widely applied in bioinformatics to solve NP-hard problems, in particular in Holy Grail areas like sequencing, multiple alignment, and structure prediction. Chapter 6 presents a selection of approximation algorithms and makes clear that the

idea of lower bounding the costs of a minimal solution, respectively of upper bounding the costs of a maximal solution, is at the heart of approximation algorithms.

Having so far presented mathematically well-founded approaches that lead to exact algorithms or at least to approximation algorithms with a provable guarantee on approximation factor, the wide field of *heuristic approaches* should not be completely omitted. It is often those surprising heuristic approaches which lead to the practically most useful solutions, seldom with an explanation why this is so and where the limits of such approaches are. In that sense, the theme of heuristics falls somehow outside the scope of a formally based book. This is reflected by the fact that we present in Chap. 7 only a small number of widely applied metaheuristics, and give only brief introductions (tutorials) to them. Having worked through these tutorials, the reader should be well prepared to read original papers or special textbooks on such heuristics - and she/he is encouraged to do so.

The book ends with a bibliography guiding the reader to a deeper study of matter.

### ***Bibliographic Remarks***

This book is clearly not intended to compete with the huge amount of excellent standard textbooks on bioinformatics, but to supplement these from the standpoint of formal problem analysis and algorithm design. Particular nice (to the personal taste of the author) introductions to the field of algorithmic bioinformatics are Clote & Backofen [20], Gusfield [31] (extensive and written in a formal and mathematically very concise manner), Pevzner [64] (a really fun to read book), Setubal & Meidanis [68], and Waterman [78] (the latter two books being truly classical bioinformatics textbooks). As almost all of the material treated in this textbook may also be found in any of these mentioned books, though sometimes these books do not present proofs or formal things up to the finest details, we almost completely omit to cite the books mentioned above (the alternative would be to steadily have citations from these books within the text). Concerning fields as complexity theory, approximation algorithms, Hidden Markov models, neural networks, or genetic algorithms we postpone recommendations for further readings to the end to the corresponding chapters. We cite original papers where the corresponding particular themes are first treated.

### ***Errata and Links***

There will be a steadily actualized errata website which can be found under [www.inf.uos.de/theo/public](http://www.inf.uos.de/theo/public). Further detected errors are always welcome under email address [sper@informatik.uni-osnabrueck.de](mailto:sper@informatik.uni-osnabrueck.de). A growing toolbox website is also available under [www.inf.uos.de/theo/public](http://www.inf.uos.de/theo/public). Here we provide for various demo tools that may serve the purpose to support understanding of algorithms presented in the book.

*Acknowledgements*

Jana Sperschneider (Albert-Ludwigs-Universität Freiburg, Germany and University of Western Australia, Australia) did a fourfold great job in the creation of this book: she did the laborious work to transform the manuscript into L<sup>A</sup>T<sub>E</sub>Xformat; she considerably improved overall presentation; she made “true English” from what I had written; she contributed all parts that cover RNA and pseudoknots. Lena Scheubert (Universität Osnabrück, Germany) contributed the chapter on genetic and ant algorithms. Thanks to many students to whom I had the opportunity to teach the themes of this book, and who improved presentations by steady criticism. Thanks also to all those students who took the opportunity to realize more or less extended implementation projects, some of which are presented on the bioinformatics toolbox website announced above. Last but not least thanks to the Springer team, in particular Hermann Engesser who supported the project from the beginning, Dorothea Glaunsinger and Frank Holzwarth who perfectly managed the production of the book.

Osnabrück, Germany,  
March 2008

*Volker Sperschneider*

Bioinformatics

Problem Solving Paradigms

Sperschneider, V.

2008, XVIII, 290 p. 208 illus., Hardcover

ISBN: 978-3-540-78505-7