Integer Linear Programming in Computational and Systems Biology

Integer linear programming (ILP) is a versatile modeling and optimization technique that is increasingly used in nontraditional ways in biology, with the potential to transform biological computation. However, few biologists know about it. This how-to and why-do text introduces ILP through the lens of computational and systems biology. It uses in-depth examples from genomics, phylogenetics, RNA and protein folding, network analysis, cancer, ecology, co-evolution, DNA sequencing and sequence analysis, pedigree and sibling inference, haplotyping, clustering, and more to establish the power of ILP. This book aims to teach the logic of modeling and solving problems with ILP, and to teach the practical "workflow" involved in using ILP in biology.

Written for a wide audience, with no biological or computational prerequisites, this book is appropriate for both entry-level and advanced courses aimed at biological and computational students, and as a source for specialists. Numerous exercises and accompanying software (in Python and Perl) demonstrate the concepts.

Dan Gusfield is Distinguished Professor of Computer Science at the University of California, Davis (UCD), and a fellow of the IEEE, the ACM, and the International Society of Computational Biology (ISCB). His previous books include *The Stable Marriage Problem* (with Robert W. Irving) and *Algorithms on Strings, Trees and Sequences*; and *ReCombinatorics*. He has served as chair of the computer science department at UCD (2000–2004), and was the founding editor-in-chief of *The IEEE/ACM Transactions of Computational Biology and Bioinformatics* until January 2009. He has been instrumental in the definition and development of the intersection between computer science and computational biology.

INTEGER LINEAR PROGRAMMING IN COMPUTATIONAL AND SYSTEMS BIOLOGY

An Entry-Level Text and Course

DAN GUSFIELD
University of California, Davis

CAMBRIDGE UNIVERSITY PRESS

CAMBRIDGE
UNIVERSITY PRESS

University Printing House, Cambridge CB2 8BS, United Kingdom

One Liberty Plaza, 20th Floor, New York, NY 10006, USA

477 Williamstown Road, Port Melbourne, VIC 3207, Australia

314–321, 3rd Floor, Plot 3, Splendor Forum, Jasola District Centre, New Delhi – 110025, India

79 Anson Road, #06–04/06, Singapore 079906

Cambridge University Press is part of the University of Cambridge.

It furthers the University's mission by disseminating knowledge in the pursuit of education, learning, and research at the highest international levels of excellence.

www.cambridge.org
Information on this title: www.cambridge.org/9781108421768
DOI: 10.1017/9781108377737

First published 2019

Printed and bound in Great Britain by Clays Ltd, Elcograf S.p.A.

A catalogue record for this publication is available from the British Library.

Library of Congress Cataloging-in-Publication Data
Names: Gusfield, Dan, author.
Title: Integer linear programming in computational and systems biology : an entry-level text and course / Dan Gusfield, University of California, Davis.
Description: Cambridge, United Kingdom ; New York, NY : Cambridge University Press, 2019. | Includes bibliographical references and index.
Identifiers: LCCN 2018061722 | ISBN 9781108421768 (hardback : alk. paper)
Subjects: LCSH: Computational biology–Mathematical models. | Systems biology–Mathematical models. | Linear programming.
Classification: LCC QH324.2 .G87 2019 | DDC 570.285–dc23
LC record available at https://lccn.loc.gov/2018061722

ISBN 978-1-108-42176-8 Hardback

Dedicated to Dick Karp, mentor and role model to me for 45 years and counting. Thank you Dick for all your wisdom, support, and friendship.

Contents

Introduction to the Book and Course

> Quoting Dick Karp, "Quoting Dan Gusfield, 'When you have an instance of a hard computational problem, try integer programming – It might work'."

I.1 WHY INTEGER PROGRAMMING?

Integer (linear) programming, abbreviated "ILP," is a versatile modeling and optimization technique that was developed for complex planning and operational decision-making. However, it has been increasingly used in *computational and systems biology* in *non traditional* ways, most importantly and inventively as a computational tool to *model* biological *phenomena*, to *analyze* biological *data*, and to *extract* biological *insight* from the models and the data. Integer programming is often (but not always) very effective in solving *instances* of hard computational problems on *realistic* biological data of current importance, despite the fact that many of those problems lack *general* algorithmic solutions that are efficient (in a provable, worst-case sense); and that the problem of solving integer programs also lacks a provable worst-case efficient algorithm.

Moreover, even for a problem where a general, worst-case efficient algorithm might be possible, the time and effort needed to find it, and the time and effort needed to implement it as a computer program, are typically much greater than the time and effort needed to formulate and implement an ILP solution to the problem. Further, it is often desirable to solve both the *minimization* and the *maximization* variants of a problem. For many problems, this requires finding and implementing two quite *different* algorithms, while an ILP solution for one variant can often be converted to an ILP formulation for the other variant, just by interchanging the words "minimize" and "maximize."

Highly engineered, commercial ILP solvers are available now (free to academics and researchers) to *solve* ILP formulations. Alternative open-source solvers are also available, and these can be effective in *some* biological applications. The improvement of the *best* solvers has been spectacular, with an estimate that (combined with faster computers) benchmark ILP problems can now be solved 200 *billion* times

faster than 25 years ago.[1] Exploiting ILP, some biological problems of importance can be modeled in a way that allows a solution in seconds on a laptop, while solutions to the more common (statistically based) models require days, weeks, or months of computation on large clusters.[2]

The effectiveness of the best ILP solvers on many problem *instances* of importance in biology opens huge opportunities. The impact of faster and easier-to-implement computation could be truly transformative in many areas of biology. However, few biologists know about integer linear programming, and when I explain that the ILP approach might translate an instance of a simple-to-describe computational problem into hundreds of thousands, or even millions, of linear inequalities that are then solved, I am often met with incredulity that such an approach could work – but it does, and often!

So, there are challenges to effectively using the ILP tools for biological problems, and educational and outreach issues that must be addressed. This book is motivated by those successes (and some failures), opportunities, and challenges. It is my attempt to reach and educate a broad range of computational biologists and students, particularly those with biology, rather than computation, backgrounds.

I.2 ABOUT THIS BOOK AND COURSE

My previous books are heavy on abstract theoretical thinking, emphasizing theorems, proofs, and methods, and are aimed at an advanced audience. Those books explicitly state that they are *not* "how-to" books. But in addition to my theoretical work, I have also done some "practical" work, particularly in my use of integer linear programming for computational biology. So in contrast to my past books, this *is* a "how-to" and a "why-do" book, and it is aimed at an audience with little or no background in either integer linear programming or computer programming.[3] The book uses meaningful examples from a wide range of topics in computational and systems biology in order to argue for the utility of integer programming, and to teach the practical "workflow" involved in using integer programming in biology.

Although the book discusses a wide variety of integer programming applications in computational and systems biology, it is *not* intended as a *comprehensive survey* of the literature, even restricted to *molecularly oriented biology*. I direct the reader to a review by G. Lancia [118], which covers much of the literature up to 2008; to deeper treatments of a narrower range of selected applications in [117] by Lancia in 2004; and to an overview of several problems [7], written by E. Althaus, G. W. Klau,

[1] However, the difference in efficiency and reliability between the best solvers and their competitors can be *enormous*, and the use of less efficient or less reliable solvers can give a very distorted impression of what is currently possible using an ILP approach.

[2] Most of the runtimes I report in this book were obtained on a Macbook Pro laptop, 2.3 GHz i7, with four processors allowing eight threads, costing under $2,000. Most of the ILP computations were done with the Gurobi Optimizer, version 6.5, from Gurobi Optimization ®. But, during the writing of the book, Gurobi released versions 7.02, 7.5, and 8.0, and some computations were done with those versions. A few computations were done using Cplex 12.6 for additional validation of the qualitative results reported in the book, and several results were obtained on an iMac i5, four processors with four threads. That processor is slightly faster than the one on the Macbook Pro, but with only four threads, the run times were similar, with the iMac at most 20% faster than the Macbook Pro. These parameters should be contrasted with typical results for computations in biology that often run for months on a cluster with hundreds to thousands of processors.

[3] However, readers with such background should still find many of the specific topics of interest.

O. Kohlbacher, H. P. Lenhof, and K. Reinert in 2009. The paper [69] also surveys several computational problems in computational biology, but the problems are first cast as *quadratic* integer programming problems, and later converted to *linear* integer programming problems.

Intended Audience My intended audience consists of computational and systems biologists, and students in biology and/or computer science/mathematics. I am particularly interested in reaching real biologists who have had little or no exposure to integer programming. Why the emphasis on biologists?

Many biologists in computational biology are excellent computer programmers, and many are very sophisticated in methods of *statistical* modeling and computation, and in many types of computational methods. For example, there are many biologists who are highly proficient with classical statistical tests and sampling, with statistical genetics, with Markov chain Monte Carlo methods, with dynamic programming, with the use of sophisticated string algorithms, with data structures, and with databases. In contrast, very few biologists are familiar with integer programming. Overwhelmingly, the papers in computational and systems biology that use integer programming are authored by researchers from computer science, engineering, mathematics, or statistics. This book, which discusses integer programming through the lens of computational and systems biology, and is written for an audience with no background in ILP, aims to change that. However, I believe that the book will also be of interest to a more advanced audience, and a source for specialists.

As a Text for a Course Part I of the book can be used as a text for an introductory course on integer linear programming in computational and systems biology.[4] All of the instructional material on integer programming, and the use of the ILP solver Gurobi Optimizer, is in Part I of the book. Additional topics from Part II can be added to match the interests of the instructor or class. I recently designed and taught such a course, the first of its kind, using material from Part I, to undergraduate students with *no* required prerequisites. About half the students were biology students, and half were from computer science. Only a few of the biology students had any background in computer programming. I believe the material in the course was quite accessible to all of the students, as most of the students did very well, particularly in their term projects.[5]

I.3 TASKS AND SKILLS

In addition to demonstrating the power of integer linear programming to model and solve instances of computational problems in biology, this book develops *skills* for four tasks:

(A) The task of developing the *solution idea* or *solution logic* for an *already well-articulated* computational problem in biology. Of course, the solution ideas must be developed with an eye toward translating the logic into an integer linear program (task B). Often, the solution logic will be straightforward or even trivial, but in a few of the problems we discuss, the logic will be more subtle.

[4] Or even for a general-purpose course on the use of integer programming.
[5] Some of these are discussed in the book.

(B) The task of creating an *abstract* ILP *formulation* for a biological problem, translating the logic developed in task A into general integer linear (in)equalities, and a linear objective function. Task B is the main focus of this book.

(C) The task of writing a *computer program* that takes in the details of an *instance* of a biological problem, and creates the *concrete* ILP formulation specifying the linear (in)equalities (coming from the result of task B) that are used to solve the given problem instance.[6]

Computer programs that accomplish task C have been written for many of the computational problems discussed in this book, and these programs can be downloaded from the book website. You do *not* need to know any computer programming to *use* those programs. But, some computer programming can be learned by examining those programs, and additionally, the book website contains a short tutorial on programming in the computer language Python, to accomplish task C. However, the book is structured so that this material is *optional*, and can be completely *skipped*. A reader can understand the material that addresses tasks A, B, and D, without any need to engage with task C.[7]

(D) The task of *using* an ILP *solver* to find an optimal solution for a concrete ILP formulation (created by task C). In this book we mainly discuss the ILP solver developed by Gurobi Optimization ®.[8] Discussion of task D is interwoven with the discussion of tasks A and B.

Biological and Mathematical Modeling There is an additional skill that is *crucial* for effective computational and systems biology. But, it is a skill that I cannot formally teach, although I can display and critique it. It is the skill of *translating* biological phenomena into *formal, mathematical models*. In this book, I address biological phenomena that are formally modeled in a way that lead to computational problems solvable by integer linear programming.

I.4 WHY LEARN FROM ME?

Of the four tasks addressed in the book, I am most proficient in tasks A and B, which most closely involve the kind of abstract, logical thinking that forms my professional training and career. But, one of the most appealing properties of integer programming is that the logical thinking required for task A is often not that demanding. And, for task B, there is a small set of *idioms* or *modeling tricks* that are very helpful in moving from a high-level solution *idea* to a collection of linear inequalities that encode that idea.

I am much less proficient in skills C and D, and I consider this a *virtue*. In this book, skill D is developed by using the ILP solver *Gurobi Optimizer*, and skill C is

[6] For small problem instances, one can write out the needed inequalities directly using a text editor or word processor, without writing a computer program. If your interest is only in learning about the power of integer programming, how to formulate ILPs, or how to use an ILP solver, small examples are sufficient. But for typical problem instances that arise in practice, you will need a computer program to *create* the concrete ILP formulations.

[7] The recent course I taught on integer programming in computational biology skipped task C entirely.

[8] Although, all of the concrete ILP formulations produced by the software can also be solved with Cplex ®, and an instructor who wants to use Cplex in place of Gurobi could easily supplement the Gurobi-specific materials in the book.

addressed with a tutorial on Python programming, posted on the book's website. Compared to true experts, I am a novice in both Python and Gurobi. I typically run Gurobi with its *default* parameters (there are over 50 adjustable parameters in Gurobi), and my Python programs usually sacrifice elegance and efficiency in order to reduce my programming effort. There are huge (thousand-plus page) books on Python, and the Gurobi documentation is equally massive. I have read and understand a *minuscule* fraction of those, and that is one of my strongest qualifications for writing this book – I don't know much about Gurobi or Python, but I know enough to get the job done. So, what I will teach you is just enough Gurobi and just enough Python to join me. True connoisseurs of Python will be disgusted by my crude Python programming and the way I explain it, and true aficionados of Gurobi will be appalled by my lack of sophistication. But sometimes ignorance is bliss, and helps to keep the work simple.

I.5 THE ORGANIZATION OF THE BOOK

I have divided the book into two parts. Both parts discuss real problems in computational and systems biology that have been, or could be, solved by integer linear programming. Most of the topics come from published research papers.

I.5.1 Part I

Part I is self-contained and can be the basis for a course on integer linear programming in computational and systems biology. All of the material I used in a recent undergraduate course on computational biology and integer programming comes from Part I. Part I starts with biological problems that can be solved with *basic* ILP techniques, and then incrementally develops and explains integer programming through biological problems that need additional, or more complex, ILP techniques. The ILP formulations in Part I are thoroughly explained, and many of the formulations are accompanied by software (available online) to generate or read problem instances, and to create the *concrete* ILP formulations to solve those instances. Using this software allows the reader to generate their own examples, and to test and extend their understanding of the topics in the book. I highly recommend that the reader do that.[9] The software also allows an instructor using this text to develop additional examples or test problems for use in the class. Part I also contains more course-related exercises than Part II, along with exercises on the practical use of Gurobi Optimizer.

I.5.2 Part II

Part II continues with additional topics in computational and systems biology that are solved by integer programming. Several of the chapters in Part II parallel chapters in Part I, further developing biological and ILP topics introduced in Part I. Most of the ILP formulations in Part II are explicitly described (as they are in Part I), but more

[9] In the class I recently taught, students made many interesting observations that were new to me, based on such explorations.

of the *explanations* for the formulations are developed in exercises. There are also more *extended exercises*, where most of a topic is developed through exercises.

Thus, Part II further exposes the power of ILP and its wide range of applications in biology, but relies on the discussions of ILP and Gurobi from Part I. Finally, (at this moment) only a few of the topics in Part II are accompanied by software, although this may change in the future, as software written by myself, or others, is added to the webpage for this book.

I.5.3 Software, Empirical Results, and Computer Programming

As mentioned earlier, many of the topics in Part I (and some in Part II) are accompanied by computer programs that read or generate data, and then create concrete ILP formulations for that data. The programs are written in the programming language Python (version 2.7), or in PERL. The reader does *not* need to know either of those languages, or even need to have any knowledge of computer programming. The reader can simply *use* those programs without needing to look inside of them. But, for the reader who wants to develop a few skills for task C, the tutorial that can be downloaded from the book website is where you can start. The URL for the book website, maintained by Cambridge Press is:

www.cambridge.org/9781108421768

The awful 13 digit number at the end is the ISBN number for the book. It is better to bookmark the page, or download all programs in a zip file, than trying to correctly remember and enter the ISBN number each time.

Empirical Results Throughout the book, I report on computational experiments I have done using Gurobi Optimizer or Cplex to solve instances of ILP formulations developed in the book. Many of these computations were done with versions of ILP solvers that are *not* the most current. The purpose of these empirical results is to *roughly* distinguish between practical and impractical computational solutions, and to illustrate the *range of practicality* of an ILP approach. Empirical results showing that an ILP approach is practical today, for a useful range of data, will remain valid as faster ILP solvers are released. So, these empirical tests are *not* intended as comprehensive, fine-grained explorations leading to advice on the best ILP solvers or the best computer architectures or the best parameters to use today for solving particular ILP problems. Periodically, papers of that type appear in the literature, often making very fine distinctions between different approaches. But those results quickly become outdated as computers, operating systems, and solvers change.

Python Programs and Programming No computer programming is needed to read this book and use the software, and the reader need never learn any. However, for a reader who wants to see details of the whole "workflow" (i.e., all tasks A through D), the tutorial on the book website introduces elements of the Python programming language. The tutorial explains the Python programs that accomplish task C for two of the biological problems discussed in the book. These programs, and all the Python programs available on the book website, are written using the *simplest* possible

elements of Python.[10] Thus, the tutorial on Python introduces just enough Python to understand those two programs, and to begin some of your own programming, but is very far from a complete tutorial on Python or computer programming.

On Figures I have drawn most of the figures in the book. Several others have been generously contributed by colleagues and students. A fair number of figures have been copied, and sometimes modified, from papers in PLoS and BMC journals, under the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), or the Creative Commons CC0 public domain dedication. A few others have been reprinted from journal papers under licenses granted by the publishers.

I.6 NO MATH

One of the prepublication reviewers of a draft of the book said that most biologists (even the ones who already want to learn about integer programming) will see the mathematical *symbols* in the book and close it fast and hard. But actually, the book contains *no* math beyond arithmetic[11] – it just looks like scary math until you read and parse it. The mathematical notation is really just a *shorthand* or a *language* for stating what is required and desired of an integer programming solution.[12] Yes, there are occasional ideas that take some time and thought to understand, and sometimes the material requires *logical puzzling* – some of which is challenging. But nothing in the book requires knowing, understanding, or learning any math you didn't already learn by the eighth grade. Of course, there is deep and beautiful mathematics *underneath* the whole field of linear programming and integer linear programming. But this book does not get into any of that – it is only about *using* integer linear programming to *model* and *solve* certain computational problems that arise in biology.

I.7 ACKNOWLEDGMENTS

Of course in writing a book, and in doing all the reading and research before that, there are many people who helped, and many people to thank. In no particular order I want to thank Bob Bixby, Ed Rothberg, Steven An, Gunnar Klau, Roded Sharan, Yufeng Wu, Chase Maguire, Thong Le, Julia Matsieva, Yelena Frid, Yun Song, Dan Brown, Russell Schwartz, Fumei Lam, Rob Gysel, Kristian Stevens, Balaji Venkatachalam, Matthias Koeppe, David Amar, Giuseppe Lancia, Carl Kingsford, Tandy Warnow, Mohammed El-Kebir, Tanya Berger-Wolf, Dick Karp, Kent Wilken, Jim Orlin, David Shmoys, Ron Shamir, Thomas Magnanti, Paola Bonizonni, Sorin Istrail, Teresa Przytycka, Sylvia Spengler, and all of the students in the fall 2017 ECS 189 class on integer programming in computational biology at UC Davis, particularly Kat Arnott for conversations on community detection, and Jonathan Kim, Parsoa Khorsand, Jessica Au, and Jessie Vuong, whose class projects I write about in this

[10] Some of the programs accompanying this book are written in PERL, and those programs are more complex. The tutorial does *not* explain the PERL programs, although they can be easily *used* by the reader.
[11] OK, maybe some very, very elementary high school algebra.
[12] We will define each piece of mathematical notation the first time it is used, and list it in the index of the book.

book; and who have generously allowed their class-project software to be distributed with this book. I also acknowledge the support of the Simons Institute for Theoretical Computing, where I wrote some of the book while on sabbatical, and the research support from the NSF grant 1528234, on integer linear programming in computational biology.

Last, but certainly not least, I want to thank my wife Carrie (again) for her continuing forbearance. In the acknowledgments of a prior book, I swore to her that it would be my last – and I lied (again).[13]

[13] But the good news is that I only have two more books I want to write.