Networked Life

How does Google sell ad space and rank webpages? How does Netflix recommend movies, and Amazon rank products? How can you influence people on Facebook and Twitter, and can you really reach anyone in six steps? Why doesn't the Internet collapse under congestion, and does it have an Achilles' heel? Why are you charged per gigabyte for mobile data, and how can Skype and BitTorrent be free? How big is the "cloud" of cloud services, and why is WiFi slower at hotspots than at home?

Driven by 20 real-world questions about our networked lives, this book explores the technology behind the multi-trillion dollar Internet, wireless and online media industries. Providing easily understandable answers for the casually curious, alongside detailed explanations for those looking for in-depth discussion, this thought-provoking book is essential reading for students in engineering, science and economics, for network industry professionals, and for anyone curious about how technological and social networks really work.

Mung Chiang is a Professor of Electrical Engineering at Princeton University, and Director of the Princeton EDGE Lab. He has received the IEEE Kiyo Tomiyasu Award, and a US Presidential Early Career Award for Scientists and Engineers, for his research on networking. A co-founder and advisor to several startups, he also received a Technology Review TR35 Award for his contributions to network technology innovation. He is a fellow of the IEEE.

Networked Life

20 Questions and Answers

MUNG CHIANG
Princeton University

CAMBRIDGE
UNIVERSITY PRESS

CAMBRIDGE UNIVERSITY PRESS
Cambridge, New York, Melbourne, Madrid, Cape Town
Singapore, São Paulo, Delhi, Mexico City

Cambridge University Press
The Edinburgh Building, Cambridge CB2 8RU, UK

Published in the United States of America by Cambridge University Press, New York

www.cambridge.org
Information on this title: www.cambridge.org/9781107024946

First published 2012

Printed in the United States by Edwards Brothers

A catalog record for this publication is available from the British Library

ISBN 978-1-107-02494-6 Hardback

Additional resources for this publication at www.cambridge.org/networkedlife

To my family

Contents

Preface

You pick up your iPhone while waiting in line at a coffee shop. You Google a not-so-famous actor and get linked to a Wikipedia entry listing his recent movies and popular YouTube clips. You check out user reviews on IMDb and pick one, download that movie on BitTorrent or stream it in Netflix. But for some reason the WiFi logo on your phone is gone and you're on 3G. Video quality starts to degrade a little, but you don't know whether it's the video server getting crowded in the cloud or the Internet is congested somewhere. In any case, it costs you $10 per gigabyte, and you decide to stop watching the movie, and instead multitask between sending tweets and calling your friend on Skype, while songs stream from iCloud to your phone. You're happy with the call quality, but get a little irritated when you see that you have no new followers on Twitter.

You've got a typical networked life, an online networked life.

And you might wonder how all these technologies "kind of" work, and why sometimes they don't. Just flip through the table of contents of this book. It's a mixture: some of these questions have well-defined formulations and clear answers while for others there is still a significant gap between the theoretical models and actual practice; a few don't even have widely accepted problem statements. This book is about formulating and answering these 20 questions.

This book is about the networking technologies we use each day as well as the fundamental ideas in the study of networks. Each question is selected not just for its relevance to our daily lives, but also for the core concepts and key methodologies in the field of networking that are illustrated by its answer. These concepts include aggregation and influence, distributed coordination, feedback control, and strategic equilibrium. And the analytic machineries are based on mathematical languages that people refer to as graph, optimization, game, and learning theories.

This is an undergraduate textbook for a new course created in 2011 at Princeton University: **Networks: Friends, Money, and Bytes**. The course targets primarily juniors and seniors in electrical engineering and computer science, but also beginning graduate students as well as students from mathematics, sciences, economics, and engineering in general. It can be viewed as a second course after the "signals and systems" course that anchors the undergraduate electrical and computer engineering curriculum today. Starting in September 2012, this course is also on free open access platforms, such as Stanford's coursera and the course's own open education website, as well as on YouTube and iTunes U.

This book weaves a diverse set of topics you would not normally see under the same cover into a coherent stream: from Arrow's impossibility and Rawls' fairness to Skype signaling and Clos networks, from collaborative filtering and firefly synchronization to MPEG/RTSP/TCP/IP and WiFi CSMA DCF. This begs a question: "So, what *is* the discipline of this book?" This is a question that most of the undergraduates do not care about. Neither does this book, which only wants to address these practical questions, using whatever modeling languages that have been observed to be the most relevant ones so far. It turns out that there is a small, common set of mathematics which we will need, but that's mostly because people have invented only a limited suite of modeling languages.

This is not a typical textbook for another reason. It does not start with general theories as do many books on these subjects, e.g., graph theory, game theory, and optimization theory, or with abstract concepts like feedback, coordination, and equilibrium. Instead it starts with concrete applications and practical answers, and sticks to them (almost) every step of the way. Theories and generalizations emerge, as if they were "accidental by-products," during the process of formulating and answering these questions.

This book can be supplemented with its website: `http://www.network20q.com`, including lecture slides, problem solutions, additional questions, examples of advanced material, further pointers to references, collections of news media coverage of the topics, "currency-earning" activities, course projects, blogs, tweets, surveys, and student-generated course material in wiki. We have created web features that turn this class into an online social network and a "networked economy."

This book can also be used by engineers, technology managers, and pretty much anyone with a keen interest in understanding how social and technological networks work. Often we sacrifice generality for accessibility, and supplement symbolic representation with numerical illustration.

- The first section of each chapter is a "short answer," and it is accessible by most people.
- Then there's a "long answer" section. If you remember differentiation and linear algebra (and occasionally a little bit of integration and basic probability), you can follow all the material there. We try to include only those symbols and equations that are really necessary to unambiguously express the ideas.
- The "examples" section contains detailed, numerical examples to reinforce the learning from the "long answer" section. On average, these first three sections of each chapter form the basis of one 80-minute lecture. Several of these lectures will go over 80 minutes while several others, including the last two, can be covered under 80 minutes.

- Each chapter concludes with a section on "advanced material," which requires the reader to be quite comfortable with symbolic operations and abstract reasoning, but can be skipped without losing the coherence and gist of the book. In the undergraduate course taught at Princeton, hardly any of the advanced material is covered. Covering all the "advanced material" sections would constitute an introductory graduate-level course. To keep the book thin, worked examples for these sections are pushed to the course website.
- At the end of each chapter, there are five homework questions, including easy drills, essential supplements, and some "out-of-syllabus" explorations about networks in biology, energy, magic, music, and transportation. The level of difficulty is indicated on a scale of one (easy) to three (hard) stars. On the course website, there is a much larger collection of additional homework problems, including many multiple-choice questions testing the basic understanding of the material.
- There are also five key references per chapter (yes, only five, in the hope that undergraduates may actually read some of these five, and my apologies to the authors of thousands of papers and books that could have been cited). These references open the door to further reading, including textbooks, research monographs, and survey articles.

This is a (relatively) thin book. It's a collage of snapshots, not an encyclopedia. It's an appetizer, not an entree. The majority of readers will not pursue a career specializing in the technical material in this book, so I take every opportunity to delete material that's very interesting to researchers but not essential to this undergraduate course. Each one of these 20 chapters deserves many books for a detailed treatment. I only highlight a few key ideas in the span of about 20 pages per chapter and 80 minutes per lecture. There are also many other mathematical languages in the study of networks, many other questions about a networked life, and many other types of networks that we do not have time to cover in one semester. But as the saying goes for a course: "It's more important to *uncover* than to cover a lot."

This is a book illustrating some pretty big ideas in networking, through 20 questions we can all relate to in our daily lives. Questions that tickle our imagination with surprises and incomplete answers. Questions that I wished I had known how to answer several years ago. Questions that are quickly becoming an essential part of modern education in electrical and computer engineering.

But above all, I hope this book is fun to read.

Mung Chiang
Princeton, NJ
July 2012

Acknowledgements

In so many ways I've been enjoying the process of writing this book and creating the new undergraduate course at Princeton University. The best part is that I got to, ironically in light of the content of this book, stay *offline* and focus on learning a few hours a day for several hundred days. I got to digest wonderful books and papers that I didn't have a chance to read before, to think about the essential points and simple structures behind the drowning sea of knowledge in my research fields, and to edit and re-edit each sentence I put down on paper. It reminded me of my own sophomore year at Stanford University one and a half decades ago. I often biked to the surreally beautiful Oval in the morning and dived into books of many kinds, most of which were not remotely related to my majors. As the saying goes, that was a pretty good approximation of paradise.

That paradise usually ends together with the college years. So I have many people to thank for granting me a precious opportunity to indulge myself again at this much later stage in life.

- The new course "Networks: Friends, Money, and Bytes" could not have been created in 2011 without the dedication of its three remarkable TAs: Jiasi Chen, Felix Wong, and Pei-yuan Wu. They did so much more for the course than a "normal" TA experience: creating examples and homework problems, initiating class activities, and running the online forum.
- Many students and postdocs in Princeton's EDGE Lab and EE Department worked with me in creating worked examples and proofreading the drafts: Chris Brinton, Amitabha Ghosh, Sangtae Ha, Joe Jiang, Carlee Joe-Wong, Yiannis Kamitsos, Haris Kremo, Chris Leberknight, Srinivas Narayana, Soumya Sen, Victoria Solomon, Arvid Wang, and Michael Wang.
- Princeton students in ELE/COS 381's first offering were brave enough to take a completely new course and contributed in many ways, not the least through the class website blogs and course projects. Students in the graduate course ELE539A also helped proofread the book draft and created multiple choice questions.
- Before I even got a chance to advertise the course, some colleagues already started planning to offer this course at their institutions in 2012: Jianwei Huang (Chinese University of Hong Kong), Hongseok Kim (Sogang University, Korea), Tian Lan (George Washington University), Walid Saad

(University of Miami), Chee Wei Tan (City University of Hong Kong), and Kevin Tang (Cornell University).

- The material in this book is inspired by discussions with colleagues in both academia and industry over the years. Since last summer, more than fifty colleagues has provided valuable suggestions directly for the course and the book. In particular, I received very detailed comments on earlier drafts of the book from Keith Cambron (AT&T Labs), Kaiser Fung (Sirius), Victor Glass (NECA), Jason Li (IAI), Jennifer Rexford (Princeton), Keith Ross (NYU Poly), Krishan Sabnani (Bell Labs), Walid Saad (University of Miami), Matthew Salganik (Princeton), Jacob Shapiro (Princeton), Kevin Tang (Cornell), and Walter Willinger (AT&T Labs), among others.
- Phil Meyler from Cambridge University Press encouraged me to turn the lecture notes into a textbook, and further connected me with a group of enthusiastic staff at CUP. I am grateful to the entire Cambridge editorial and marketing team across their UK and New York offices.
- This course was in part supported by a grant from the US National Science Foundation, in a program run by Darleen Fisher, for a team consisting of two engineers and two social scientists at Princeton. I'm glad to report that we achieved this educational goal in our proposal, and did that before the project's official end date.

My appreciation traces back to many of my teachers. For example, I've had the fortune to be co-advised in my Ph.D. study by Stephen Boyd and Tom Cover, two brilliant scholars who are also superb teachers. Their graduate-level textbooks, *Convex Optimization* by Boyd and Vandenberghe and *Elements of Information Theory* by Cover and Thomas, are two towering achievements in engineering education. Read these two books, and you'll "experience" the definition of "clarity," "accessibility," and "insight." When I was writing research papers with them, Tom would spend many iterations just to get one notation right, and Stephen would even pick out each and every LaTex inconsistency. It was a privilege to see first-hand how the masters established the benchmarks of technical writing.

Stephen and Tom were also the most effective lecturers in classroom, as was Paul Cohen, from whom I took a math course in my sophomore year. Pulling off the sweatshirt and writing with passion on the blackboard from the first moment he entered the classroom, Paul could put your breath on hold for 80 minutes. Even better, he forgot to give us a midterm and then gave a week-long, take-home final that the whole class couldn't solve. He made himself available for office hours on-demand to talk about pretty much anything related to math. The course was supposed to be on PDE. He spent just four lectures on that, and then introduced us to eighteen different topics that quarter. I've forgotten most of what I learned in that course, but I'll always remember that learning can be so much fun.

In the same winter quarter that I took Stephen's and Tom's courses, I also took from Richard Rorty a unique course at Stanford called "From Religion through Philosophy to Literature," which pulled me out of Platonism to which I had been increasingly attached as a teenager. Talking to Rorty drastically sharpened my appreciation of the pitfalls of mistaking representations for reality. A side-benefit of that awakening was a repositioning of my philosophy of science, which propagated to the undercurrents of this book.

Three more inspirations, from those I never met:

- Out of all the biographies I've read, the shortest one, by far, is by Paul Johnson on Churchill. And it's by far the most impactful one. Brevity is power.
- But even a short book feels infinitely long to the author until it goes to the press. What prevented me from getting paralyzed by procrastination is Frederick Terman's approach of writing textbooks while leading a much busier life (serving as a Dean and then the Provost at Stanford, and creating the whole Silicon Valley model): write one page each day.
- Almost exactly one century ago, my great grandfather, together with his brother, wrote some of the first modern textbooks in China on algebra and on astronomy. (And three decades ago, my grandfather wrote a textbook on econometrics at the age of seventy.) As I was writing this book, sometimes I couldn't help but picture the days and nights that they spent writing theirs.

For some reason, the many time commitments of a professor are often hard to compress. And I couldn't afford to cut back on sleep too much, for otherwise the number of mistakes and typos in this book would have been even larger. So it's probably fair to say that each hour I spent writing this book has been an hour of family time lost. Has that been a good tradeoff? Definitely not. So I'm glad that the book is done, and I'm grateful to my family for making that happen: my parents who helped take care of my toddler daughter when I was off to dwell in this book, my wife who supported me sitting there staring at my study's ceiling despite her more important job of curing the ill, and Novia who could have played with her Daddy a lot more in the past year. This book was written with my pen and their time.

Roadmap

This roadmap is written for course instructors, or perhaps as an *epilogue* for students who have already finished reading the book, especially Figure 0.3 and the list of 20 ideas below it. It starts with a taxonomy of the book and introduces its organization and notation. Then it discusses the similarities and differences between this book and some excellent books published over the last decade. Then it highlights three pedagogical principles guiding the book: Just In Time, Bridge Theory and Practice, and Book As a Network, and two contexts: the importance of domain-specific functionalities in network science and the need for undergraduate-curriculum evolution in electrical and computer engineering. It concludes with anecdotes of arranging this course as a social and economic network itself.

Taxonomy and Organization

The target audience of this book is both students and engineering professionals. For students, the primary audience are those from engineering, science, economics, operations research, and applied mathematics, but also those on the quantitative side of sociology and psychology.

There are three ways to use this book as a textbook.

- *An undergraduate general course at sophomore or junior level*: Go through all 20 chapters without reading the Advanced Material sections. This course serves as an introduction to networks before going further into senior-level courses in four possible directions: computer networking, wireless communication, social networks, or network economics.
- *An undergraduate specialized course at senior level*: Pick either the social and economic network chapters or the technology and economic network chapters, and go through the Advanced Material sections in those chapters.
- *A first-year graduate level course*: Go through all 20 chapters, including the Advanced Material sections.

While this book consists of 20 chapters, there are just four key recurring concepts underlying this array of topics. Table 0.1 summarizes the mapping from chapter number to the concept it illustrates.

Table 0.1 Key concepts: The chapters where each of the four key concepts show up for different types of networks.

Network Type	Aggregation & Influence	Distributed Coordination	Feedback Control	Strategic Equilibrium
Wireless		1	19	
Internet		10, 13, 16	14	
Content Distribution		15, 17	18	
Web	3, 4, 5			2
Online Social	6,8	9	7	
Internet Economics		20		11, 12

The modeling languages and analysis machineries originate from quite a few fields in applied mathematics, especially the four foundations summarized in Table 0.2.

Table 0.2 Main methodologies: The chapters where each of the four families of mathematical languages are used in different types of networks.

Network Type	Graph Theory	Optimization Theory	Game Theory	Learning Theory
Wireless		18, 19	1	
Internet	10	13, 14, 16		
Content Distribution		15, 17		
Web	3		2	4, 5
Online Social	7, 8, 9		6	
Internet Economics		11	20	12

The order of appearance of these 20 questions is arranged so that clusters of highly related topics appear next to each other. Therefore, we recommend going through the chapters in this sequence, unless you're OK with flipping back every now and then when key concepts from prior chapters are referenced. Figure 0.1 summarizes the "prerequisite" relationship among the chapters.

This book cuts across both networks among devices and networks among people. We examine networks among people that overlay on top of networks among devices, but also spend half of the book on wireless networks, content distribution networks, and the Internet itself. We'll illustrate important ideas and useful methodologies across both types of networks. We'll see striking parallels in the underlying analytic models, but also crucial differences due to domain-specific details.

We can also classify the 20 questions into three groups in terms of the stages of development in formulating and answering them:

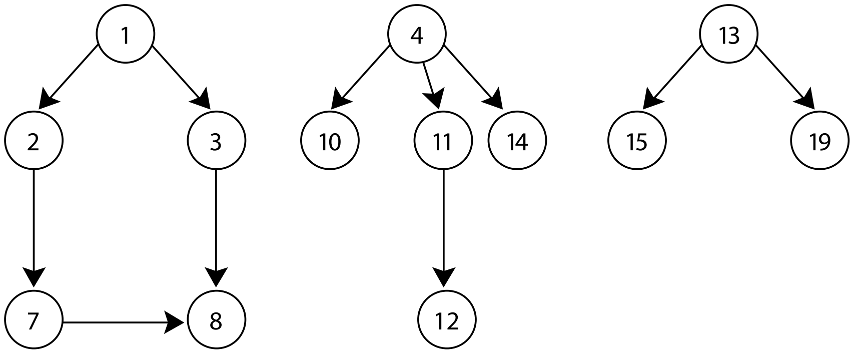

Figure 0.1 Dependence of mathematical background across some of the chapters is shown in these graphs. Each node is a chapter. Each directional link is a dependence relationship, e.g., Chapter 8's material requires that in Chapter 3 (which in turn requires that in Chapter 1) and that in Chapter 7 (which in turn requires that in Chapter 2, which in turn requires that in Chapter 1). Chapters 1, 4, and 13, the root nodes of these three trees, offer foundational material for ten other chapters. Some chapters aren't shown here because they don't form part of a dependence tree.

- Question well formulated, and theory-inspired answers adopted in practice: 1, 2, 3 4, 9, 10, 11, 13 14, 15, 16, 17, 18, 19.
- Question well formulated, but there's a gap between theory and practice (and we will discuss some possible bridges over the gaps): 12, 20.
- Question less well formulated (but certainly important to raise and explore): 5, 6, 7, 8.

It's comforting to see that most of our 20 chapters belong to the first group. Not surprisingly, questions about technological networks tend to belong to the first group, with those about social and economic networks gravitating more towards the second and third groups. It's often easier to model networked devices than networked human beings with predictive power.

Not all chapters explicitly study the impact of network topology, e.g., Chapter 7 studies influence models with decision externalities that are based on population sizes, while Chapter 8 looks at influence models with topology taken into account.

A quick word about the homework problems. There are five problems at the end of each chapter. These are a mixture of easy drills, simple extensions, challenging mini-research projects, and open-ended questions. Some important topics that we cannot readily fit into the main flow of the text are also postponed to the homework problem section. For those looking for more of the easy drills, the course website `www.network20q.com` offers additional questions.

Notation

We use **boldface** text for key terms when each is first defined. We use *italics* to highlight important, subtle, or potentially confusing points.

We use boldface math symbols to denote vectors or matrices, e.g., $\mathbf{x}$, $\mathbf{A}$. Vectors are column vectors by default. We do not use special fonts to represent sets when they are clear from the context. We use (t) to index iterations over continuous time, and $[t]$ or $[k]$ to index iterations over discrete time. We use * to denote optimal or equilibrium quantities.

Some symbols have different meanings in different chapters, because they are the standard notation in different communities.

Related Books and Courses

There's no shortage of books on networks of many kinds. The popular ones that appeared in the past decade fall into two main groups:

- Popular science books, many of them filled with historical stories, empirical evidence, and sometimes a non-mathematical sketch of technical content. Some of the widely-read ones are *Bursts, Connected, Linked, Money Lab, Planet Google, Six Degrees, Sync, The Perfect Storm, The Tipping Point*, and *The Wisdom of Crowds*. Two other books, while not exactly on networks, provide important insights into many topics in networking: *Thinking, Fast and Slow* and *The Black Swan*. On the technology networks side, there are plenty of "for dummies" books, industry certification prep books, and entrepreneurship books. There are also several history-of-technology books, e.g., *Where the Geeks Stay Up Late* and *The Qualcomm Equation.*
- Popular undergraduate- or graduate-level textbooks. On the graph-theoretic and economic side of networking, three excellent textbooks appeared in 2010: *Networks, Crowds, and Markets* by Easley and Kleinberg, *Networks* by Newman, and *Social and Economic Networks* by Jackson. The last two are more on the graduate level. An earlier popular textbook is *Social Network Analysis: Methods and Applications* by Wasserman and Faust. On the computer networking side, there's a plethora of excellent textbooks. Two particularly popular ones today are *Computer Networking: A Top-Down Approach* by Kurose and Ross, and *Computer Networks: A Systems Approach* by Peterson and Davie. On wireless communications, several textbooks published in the last few years have become popular: *Wireless Communications* by Molisch, *Wireless Communications* by Goldsmith, and *Fundamentals of Wireless Communication* by Tse and Viswanath.

As illustrated in Figure 0.2, this book fills in the gap between existing groups of books.

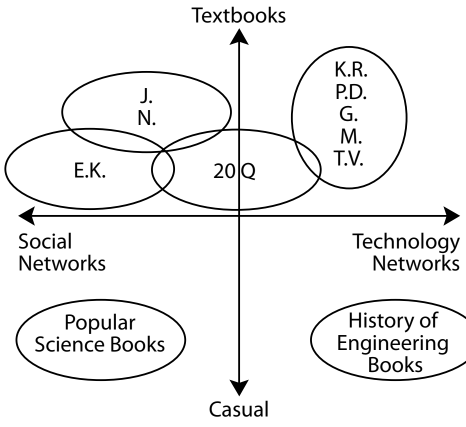

Figure 0.2 A cartoon illustrating roughly where some of the related books sit on two axes: one on the level of difficulty ranging from leisurely reading to graduate-level textbooks, and another on the mix of topics ranging from social and economic networks to technological networks. E.K. stands for Easley and Kleinberg, J. stands for Jackson, N. stands for Newman, K. R. stands for Kurose and Ross, P. D. stands for Peterson and Davie, G stands for Goldsmith, M stands for Molisch, and T.V. stands for Tse and Viswanath. 20Q stands for this book.

- Each chapter is driven by a practical question or observation, and the answers (or approximate answers) are explained using the rigorous language of mathematics. But mathematics never precedes practical problems.
- It also maintains a balance between social/economic networks and Internet/wireless networks, and between graph/economic theory and optimization/learning theory. For example, why WiFi works slower in hot spots is given as much attention as how Google auctions its ad spaces, and how IPTV networks operate is given as much detail as when information cascades initiate in a social group.
- A main goal of this book is to put social economic networks and technological networks side by side, and highlight their surprising similarities in spirit and subtle differences in detail. These examples range from the relation between Qualcomm's CDMA power control and Google's PageRank webpage ranking, to the connection between Galton's ox-weight estimation and 802.11n multiple-antenna WiFi.

These are also the differences between the Princeton undergraduate course and the seminal courses by Easley and Kleinberg at Cornell, and by Kearns at Penn. Those two courses have inspired a few similar courses at the interface between economics and computer science, such as those by by Acemoglu and Ozdaglar at MIT, by Chaintreau at Columbia, by Kempe at USC, by Parkes at Harvard, by Prabhakar at Stanford, by Spielman at Yale, by Wierman at Caltech...

These excellent courses have started structuring social and economic networking topics to undergraduates. On the other hand, both computer networking and wireless communications courses are standard, indeed often required, courses at many universities' Computer Science (CS) and Electrical Engineering (EE) departments. And across the EE, CS, Economics, Operations Research, and Applied Math departments, optimization theory, game theory, learning theory, and graph theory all have their separate courses. The new course at Princeton sits in-between the CS/Econ topics and the EE topics on networks. We hope there'll be more courses in EE and CS departments around the world that use unambigious languages to teach the concepts and methods common to social, economic, and technological networks.

Pedagogical Principles

This book and the associated course are also an experiment in three principles of teaching networks: JIT, BTP, and BAN.

Principle 1: Just In Time (JIT)

Models are often crippled by their own assumptions to start with, and frequently end up being largely irrelevant to what they set out to enable. Once in a while this is not true, but that's a low-probability event. That's why modeling is hard, especially for networks. So, before presenting any model, we first try to justify why the models are really necessary for the practical problems we face in each chapter. The material is arranged so that extensive mathematical machinery is introduced bit by bit, each bit presented just in time for the question raised. We enforce this "just-in-time" policy pretty strictly: no mathematical machinery is introduced unless it's used within the same section.

This might seem to be a rather unconventional way to write a textbook on the mathematical side of engineering. Usually a textbook asks the students to be patient with 50, 100, sometimes 200 pages of mathematics to lay the foundation first, and promises that motivating applications are coming after these pages. It's like asking a three-year-old to be patient for a long drive and promising ice-cream cones after many miles on the highway. In contrast, this book hands out an ice-cream cone every minute along the way, so that the three-year-old becomes very motivated to keep the journey going. It's more fun when gratification isn't delayed. "Fun right now" and "instant gratification" are what this book tries to achieve.

This book is an experiment motivated by this hypothesis: what professors call "fundamental knowledge" can be taught as "by-products" in the answers to practical questions that the undergraduates are interested in. A devoted sequence of lectures focusing exclusively (or predominantly) on the fundamental knowledge is not the *only* way to teach the material. Maybe we could also chop up the material and sprinkle it around. This does not “water-down” the material, it simply reorganizes it so that it shows up right next to the applications in each and every lecture. The downside is that the standard trains of thought running through the mathematical foundation of research communities are interrupted many times. This often leaves me feeling weird because I could not finish my normal teaching sequence, but the instructor feeling uncomfortable is probably a good sign. The upside is that undergraduates, who may not even be interested in a career in this field, view the course as completely driven by practical questions.

For example, the methodologies of optimization theory are introduced bit by bit in this book: linear programming and Perron-Frobenius theory in power control, convexity and least squares in Netflix recommendation, network utility maximization in Internet pricing, dynamic programming and multi-commodity flow in Internet routing, the gradient algorithm and dual decomposition in congestion control, and combinatorial optimization in peer-to-peer networks.

The methodologies of game theory are introduced bit by bit: the basic definitions in power control, auction theory in ad-space auctions, bargaining theory in Wikipedia consensus formation as well as in two-sided pricing of Internet access, and selfish maximization in tipping.

The methodologies of graph theory are introduced bit by bit: matching in ad-space auctions, consistency and PageRank in Google search, bipartite graph in Netflix recommendation, centrality, betweenness, and clustering measures in influence models, small worlds in social search, scale-free graphs in Internet topology, the Bellman–Ford algorithm and max flow min cut in Internet routing, and tree embedding in peer-to-peer networks.

The methodologies of learning theory are introduced bit by bit: collaborative filtering in Netflix recommendation, Bayesian estimation and adaptive boosting in ratings, and community detection in influence models.

Principle 2: BTP (Bridge Theory and Practice)

The size of the global industry touched upon by these 20 questions is many trillions of dollars. Just the market capitalizations of the 20 most relevant US companies to this book: Apple, Amazon, AT&T, Cisco, Comcast, Disney, eBay, EMC, Ericsson, Facebook, Google (including YouTube), Groupon, HP, Intel, LinkedIn, Microsoft (including Skype), Netflix, Qualcomm, Verizon, and Twitter added up to over $2.22 trillion as of July 4, 2012.

In theory, this book’s theories are directly connected to the practice in this multi-trillion-dollar industry. In practice, that’s not always true, especially in fields like networking where stable models, like the additive Gaussian noise channel for copper wire in communication theory, often do not exist.

Nonetheless, we try to strike a balance between

- presenting enough detail so that answers to these practical questions are grounded in actual practice rather than in “spherical cows” and “infinite planes," (although we couldn't help but keep "rational human beings" in several chapters), and

- avoiding too much detail that reduces the "signal-noise-ratio" in illustrating the fundamental principles.

This balance is demonstrated in the level of detail with which we treat network protocol descriptions, Wikipedia policies, Netflix recommendation algorithms, etc. And this tradeoff explains the (near) absence of random graph theory and of Internet protocols' header formats, two very popular sets of material in standard textbooks in math/CS-theory/sociology and in CS-systems/EE curricula, respectively.

Some of these 20 questions are currently trapped in particularly deep theory–practice gaps, especially those hard-to-formulate questions in Chapters 5 and 6, and those hard-to-falsify answers in Chapters 7 and 8. The network economics material in Chapters 11 and 12 also fits many of the jokes about economists, too many to quote here. (A good source of them is Taleb's *The Bed of Procrustes.*) Reverse engineering, shown across several chapters, has its own share of accurate jokes: "Normal people look at something that works in theory, and wonder if it'll also work in practice. Theoreticians look at something that works in practice, and wonder if it'll also work in (their) theory."

Time and time again, we skip the techniques of mathematical acrobats, and instead highlight the never-ending struggles between representations and realities during modeling: the process of "mathematical crystallization" where (most) parts of reality are thrown out of the window so that what remains becomes tractable using today's analytic machineries. What often remains unclear is whether the resulting answerable questions are still relevant and the resulting tractable models still have predictive powers. However, when modeling is done "right," engineering artifacts can be explained rather than just described, and better design can be carried out top-down rather than by "debug and tweak" It's often been quoted (mostly by theoreticians like me) that "there's nothing more practical than a good theory," and that "a good theory is the first-order exponent in the Taylor expansion of reality." Perhaps these can be interpreted as *definitions* of what constitutes a "good" theory. By such a definition, this book has traces of good theory, thanks to many researchers and practitioners who have been working hard on bridging the theory-practice gaps in networking.

Principle 3: BAN (Book As a Network)

Throughout the chapters, comparisons are constantly drawn with other chapters. This book itself is a network, a network of ideas living in nodes called chapters, and we grasp every opportunity to highlight each possible link between these nodes. The most interesting part of this book is perhaps this network effect among ideas: to see how curiously related, and yet crucially different they are.

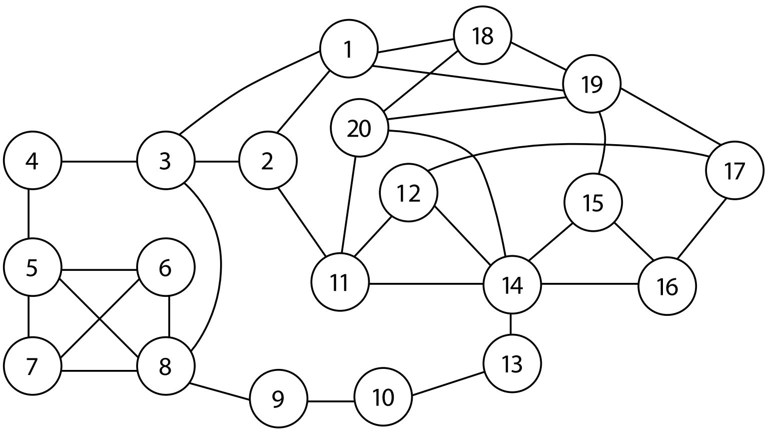

Figure 0.3 Intellectual connections across the chapters. Each node is a chapter, and each bidirectional link is an intellectual connection, via either similar concepts or common methodologies. Cliques of nodes and multiple paths from one node to another are particularly interesting to observe in this graph.

Figure 0.3 shows the main connections among the chapters. This is what the book is about: weave a network of ideas (about networks), and the positive network effect comes out of that.

We can extract the top 20 ideas across the chapters. The first 10 are features of networks, the next 5 design ideas, and the last 5 modeling approaches.

1. Resource sharing (such as statistical multiplexing and fairness): Chapters 1, 11, 13, 14, 15, 16, 17, 18, 20.
2. Opinion aggregation and consensus formation: Chapters 3, 4, 5, 6, 18.
3. Positive network effect (such as resource pooling and economy of scale): Chapters 9, 11, 13, 15, 16.
4. Negative network effect (such as tragedy of the commons): Chapters 11, 20.
5. The wisdom of crowds (diversity gain and efficiency gain): Chapters 7, 8, 18, 19.
6. The fallacy of crowds (cascade and contagion): Chapters 7, 8.
7. Functional hierarchy and layering: Chapters 13, 14, 15, 17, 19.
8. Spatial hierarchy and overlaying: Chapters 10, 13, 15, 16, 17.
9. From local actions to global property: Chapters 1, 6, 7, 8, 13, 14, 15, 18.
10. Overprovision capacity vs. overprovision connectivity: Chapters 14, 15, 16.
11. Feedback control: Chapters 1, 7, 13, 14.
12. Utility maximization: Chapters 1, 2, 11, 12, 14, 20.
13. Protocols: Chapters 14, 15, 17, 19.
14. Signaling: Chapters 6, 19.
15. Randomization: Chapters 3, 15, 18.
16. Graph consistency models: Chapters 3, 13.

17. Strategic equilibrium models: Chapters 1, 2, 15.
18. Generative model (and reverse engineering): Chapters 9, 10, 14.
19. Latent-factor models: Chapter 4.
20. Axiomatization models: Chapters 6, 20.

In the first offering of this course at Princeton, the undergrads voted (by Borda count) "resource sharing," "opinion aggregation," and "positive network effect" as the top three concepts they found most useful. They also voted the key equations in PageRank, distributed power control, and Bellman–Ford as the top three equations.

Almost every one of these 20 ideas cuts across social/economic networks and technological networks. Some examples are given below.

- The emergence of global coordination through local actions based on local views is a recurring theme, from influence models in social networks to routing and congestion control in the Internet, and from consumer reaction to pricing signals to power control in wireless networks.
- Resource sharing models, in the form of additive sharing $x + y \leq 1$, or multiplicative sharing $x/y \geq 1$, or binary sharing $x, y \in \{0, 1\}, x + y \leq 1$, are introduced for network pricing as well as the classic problems of congestion control, power control, and contention control.
- The (positive) network effect is often highlighted in social and economic networks. It also finds a concrete realization in how content is shared over the Internet through peer-to-peer protocols and how data centers are scaled up.
- "The wisdom of (independent and unbiased) crowds" is another common theme. There are two types of "wisdom" here. (1) Diversity gain in reducing the chance of some bad event (typically represented mathematically by $1 - (1 - p)^N$, where N is the size of the crowd and p the probability of some bad event). (2) Efficiency gain in smoothing out some average metric (typically represented mathematically as a factor-N in the metric). Both types are observed in social networks and in the latest generation of 4G and WiFi wireless networks.
- Consensus formation is used in computing webpage importance scores in PageRank as well as in discovering the right time to transmit in WiFi.
- Spatial hierarchy is used both in how search is done in a small world and in how the Internet is structured.
- The design methodology of feedback control is used in influence models in social networks and congestion control in the Internet.
- Utility maximization is used in auctioning advertising spots and setting Internet access pricing.
- The power method is used both in Google's PageRank and in Qualcomm's distributed power control.
- Randomization is used in PageRank and 802.11 CSMA.
- Strategic equilibrium models are used in auctioning and BitTorrent.

- Reverse engineering is used in studying scale-free networks and TCP.
- Axiomatization is used in voting procedure and fairness evaluation.

This list goes on. Yet equally important are the subtle differences between technological and socio-economic networks. Exhibit A for this alert is the (non-existence of) the Achilles' heel of the Internet and the debate between two generative models (preferential attachment vs. constrained optimization) of scale-free networks.

Two Bigger Pictures

There are two broader themes in the backdrop of this book:

- *Instill domain-specific functionalities to a generic network science.* A "network science" around these 20 questions must be based on domain-specific models and on the pursuit of falsification. For example, while a random graph is elegant, it's often neither a relevant approach to design nor the only generative model to explain what we see in this book. And as much as metrics of a static graph are important, engineering protocols governing the *functionalities* of feedback, coordination, and robustness are just as crucial as the *topological* properties of the graph like the degree distribution.
- *Revisit the Electrical and Computer Engineering (ECE) undergraduate curriculum.* In the standard curriculum in ECE since around the 1960s, a "signals and systems" course is one of the first foundational courses. As networks of various kinds play an increasingly important role both in engineering design and in the society, it's time to capture fundamental concepts in networking in a second systems course. Just as linear time-invariant systems, sampling, integral transforms, and filter design have laid the foundation of ECE curriculum since the 1960s, we think the following concepts have now become fundamental to teach to future ECE students (whether they are taught in the JIT way or not): patterns of connections among nodes, modularization and hierarchy in networked systems, consistency and consensus in graphs, distributed coordination by pricing feedback, strategic equilibrium in competition and cooperation, pros and cons of scaling up, etc.

  So this book is an experiment in both *what* to teach and *how* to teach in an ECE undergraduate course in systems: what constitutes core knowledge that needs to be taught and how to teach it in a context that enhances learning efficiency. As much as we appreciate FIR and IIR filter design, Laplace and Z transforms, etc., maybe it's about time to explore the possibility of reducing the coverage of these topics by just a tiny bit to make room for mathematical notions just as fundamental to engineering today. And we believe the best way to drive home these ideas is to tie in with applications that teenagers, and many of the older folks, use every day.

Class as a Social and Economic Network

The class "Networks: Friends, Money, and Bytes," created in parallel to this book in Fall 2011 and cross-listed in EE and CS at Princeton University, was a social and economic network itself. We tweeted, we blogged, and we created wikis. On the first day of the class, we drew a class social graph, where each node is a student, and a link represents a "know by first name before coming to the first lecture" relationship. After the last lecture, we drew the graph again.

We also created our own currency called "nuggets." The TAs and I "printed" our money as we saw fit. There were several standard ways to earn nuggets, including catching typos in lecture notes and writing popular blogs. There were ten class activities beyond homework problems that were rewarded by nuggets, including one activity in which the students were allowed to buy and sell their homework solutions using auctions. The matching of students and class project topics was also run through bidding with nuggets. Eventually the nugget balances translate into an upward adjustment of grades. To see some of the fun of ELE/COS 381 at Princeton University, visit `www.network20q.com`.

Starting in Fall 2012, this course is offered on Stanford's coursera and its own open education course website, and on YouTube and iTunes U too. The Princeton offering adopts the approach of flipped classroom advocated by the Khan Academy. With many parallel, on-going efforts from different universities and companies (CodeAcademy, coursera, EdX, udacity, etc.), it will take a few years before the landscape of open online education becomes stable. Higher education will be enhanced by this movement that has been gathering momentum since MIT's open courseware initiative in 2002. Yet many issues remain to settle at the time of writing this book: the modes and efficiency of students' learning, the boundary and reach of higher education, the prioritization and value propositions of a university's missions, the roles of faculty and the nature of classroom teaching, the differentiation between self-education and branded-certification, the authenticity and ownership of student-activity records, the tuition revenue to universities and business models of open access platforms, the drawbacks of monetization on for-profit platforms... What is already clear, however, is that this mode of education cannot be feasible without the widespread use of mobile data devices, the video-watching habit of the YouTube generation, or the crowd-sourcing of social-networked online study group. These are exactly some of the key topics studied in this course. From voting of popular questions to distributing video over 4G networks, this is a course *about* these networks and taught *through* these networks.

1 What makes CDMA work for my smartphone?

1.1 A Short Answer

Take a look at your iPhone, Android phone, or a smartphone running on some other operating system. It embodies a remarkable story of technology innovations. The rise of wireless networks, the Internet, and the web over the last five decades, coupled with advances in chip design, touchscreen material, battery packaging, software systems, business models... led to this amazing device you are holding in your hand. It symbolizes our age of networked life.

These phones have become the mobile, lightweight, smart centers of focus in our lives. They are used not just for voice calls, but also for **data applications**: texting, emailing, browsing the web, streaming videos, downloading books, uploading photos, playing games, or video-conferencing friends. The throughputs of these applications are measured in bits per second (bps). These data fly through a **cellular network** and the **Internet**. The cellular network in turn consists of the radio air-interface and the core network. We focus on the air-interface part in this chapter, and turn to the cellular core network in Chapter 19.

Terrestrial wireless communication started back in the 1940s, and cellular networks have gone through generations of evolution since the 1970s, moving into what we hear as 4G these days. Back in the 1980s, some estimated that there would be 1 million cellular users in the USA by 2000. That turned out to be one of those way-off under-estimates that did not even get close to the actual impact of networking technologies.

Over more than three decades of evolution, a fundamental concept of cellular architecture has remained essentially the same. The entire space of deployment is divided into smaller regions called **cells**, which are often represented by hexagons as in Figure 1.1, thus the name cellular networks and cell phones. There is one **base station** (BS) in each cell, connected on the one side to switches in the core network, and on the other side to the **mobile stations** (MSs) assigned to this cell. An MS could be a smart phone, a tablet, a laptop with a dongle, or any device with antennas that can transmit and receive in the right frequencies following a cellular network standard. There are a few other names for them, for example, sometimes an MS is also called a User Equipment (UE) and a BS called a Node B (NB) or an evolved Node B (eNB).

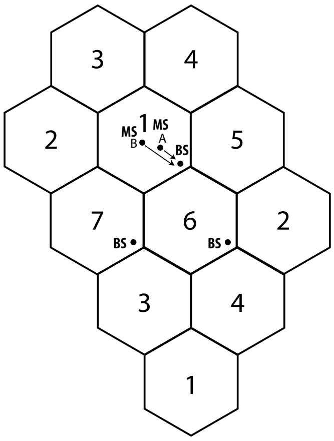

Figure 1.1 Part of a typical cellular network with a frequency reuse of 7. Each cell is a hexagon with a base station (BS) and multiple mobile stations (MSs). Only a few of them are drawn in the figure. Each BS has three directional antennas, each of which covers a 120-degree sector. Some mobile stations, like MS A, are close to the base station with strong channels to the BS. Others, like MS B, are on the cell edge with weak channels. Attenuation enables frequency reuse, but the variability of and the inability to control attenuation pose challenges to wireless cellular network design.

We see a clear hierarchy, a fixed infrastructure, and one-hop radio links in cellular networks. This is in contrast to other types of wireless networks. Moreover, the deployment of base stations is based on careful radio engineering and tightly controlled by a wireless provider, in contrast to WiFi networks in Chapter 18.

Why do we divide the space into smaller regions? Because the wireless spectrum is scarce and radio signals weaken over space.

Transmitting signals over the air means emitting energy over different parts of the electromagnetic **spectrum**. Certain regions of the spectrum are allocated by different countries to cellular communications, just like other parts of the spectrum are allocated to AM and FM radio. For example, the 900 MHz range is allocated for the most popular 2G standard called GSM, and in Europe the 1.95 GHz range and 2.15 GHz range are allocated for UMTS, a version of the 3G standard. Some part of the spectrum is unlicensed, like in WiFi, as we will see in Chapter 18. Other parts are licensed, like those for cellular networks, and a wireless service provider needs to purchase these limited resources with hefty prices. The spectrum for cellular networks is further divided into chunks, since it is often easier for transmitters and receivers to work with narrower frequency bands, e.g., on the order of 10 MHz in 3G.

The signals sent in the air become weaker as they travel over longer distances. The amount of this attenuation is often proportional to the square, or even the fourth power, of the distance traversed. So, in a typical cellular network, the signals become too weak to be accurately detected after a couple of miles. At

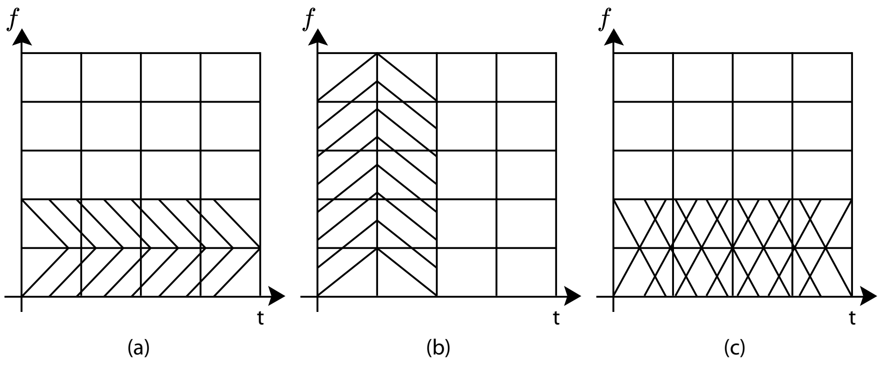

Figure 1.2 Part of a time–frequency grid is shown in each graph. For visual clarity, we only show two slices of resources being used. (a) FDMA and (b) TDMA are dedicated resource allocation: each frequency band or timeslot is given to a user. In contrast, (c) CDMA is shared resource allocation: each time–frequency bin is shared by multiple users, all transmitting and receiving over the same frequency band and at the same time. These users are differentiated by signal processing. Power control also helps differentiate their signals.

first glance, this may sound like bad news. But it also means that the frequency band used by base station A can be reused by another base station B sufficiently far away from A. All we need to do is tessellate the frequency bands, as illustrated in Figure 1.1, so that no two cells share the same frequency band if they are too close. In Figure 1.1, we say that there is a **frequency reuse factor** of 7, since we need that many frequency bands in order to avoid having two close cells sharing the same frequency band. **Cellular architecture** enables the network to scale up over space. We will visit several other ways to scale up a network later.

Now, how can the users in the *same* cell share the same frequency band? There are two main approaches: orthogonal and non-orthogonal allocation of resources.

Frequency is clearly one type of resource, and time is another. In **orthogonal allocation**, each user is given a small band of frequency in Frequency-Division Multiple Access (**FDMA**), or a timeslot in Time-Division Multiple Access (**TDMA**). Each user's allocation is distinct from others, as shown in Figure 1.2(a) and (b). This often leads to an inefficient use of resources. We will see in later chapters a recurring theme: a dedicated assignment of resources to users becomes inefficient when users come and go frequently.

The alternative, **non-orthogonal allocation**, allows all users to transmit at the same time over the same frequency band, as in Code-Division Multiple Access. **CDMA** went through many ups and downs with technology adoption from 1989 to 1995, but is now found in all the 3G cellular standards as part of the design. In CDMA's first standard, IS-95 in the 2G family, the same frequency band is reused in all the cells, as illustrated in Figure 1.2(c). But how can we distinguish the users if their signals overlap with each other?

Think of a cocktail party with many pairs of people trying to carry out individual conversations. If each pair takes turns in communicating, and only one person gets to talk during each timeslot, we have a TDMA system. If all pairs can communicate at the same time, and each uses a different language to avoid confusion, we have a CDMA system. But there are not enough languages whose pronunciations do not cause confusion, and human ears are not that good at decoding, so interference is still an issue.

How about controlling each person's volume? Each transmitter adjusts the volume of its voice according to the relative distances among the speakers and listeners. In a real cocktail party, unless there is some politeness protocol or it hurts people's vocal chord to raise their voice, we end up in a situation where everyone is shouting and yet most people cannot hear well. Transmit power control should mitigate this problem.

The core idea behind the CDMA standards follows our intuition about the cocktail party. First, the transmitter multiplies the digital signals by a sequence of 1s and minus 1s, a sequence we call the **spreading code**. The receiver multiplies the received bits by the same spreading code to recover the original signals. This is straightforward to see: 1×1 is 1, and -1×-1 is also 1. What is non-trivial is that a family of spreading codes can be designed such that only *one* spreading code, the original one used by the transmitter, can recover the signals. If you use any other spreading code in this family, you will get noise-like, meaningless bits. We call this a family of **orthogonal codes**. Users are still separated by orthogonalization, just along the "code dimension" as opposed to the more intuitive "time dimension" and "frequency dimension." This procedure is called direct sequence **spread spectrum**, one of the standard ways to enable CDMA.

However, there may not be enough orthogonal spreading codes for all the mobile stations. Families of orthogonal codes are limited in their sizes. Furthermore, a slight shift on the time axis can scramble the recovered bits at the receiver. We need the clocks on all the devices to be synchronized. But this is infeasible for the **uplink**, where mobiles talk to the base station: MSs cannot easily coordinate their clocks. It is difficult even in the **downlink**, where the base station talks to the mobiles: the BS has a single clock but the wireless channel distorts the bits. Either way, we do not have perfectly orthogonal spreading codes, even though these imperfect codes still provide significant "coding gain" in differentiating the signals.

We need an alternative mechanism to differentiate the users and to tackle the **interference** problem. Wireless signals are just energy propagating in the air, and one user's signal is every other user's interference. Interference, together with the attenuation of signals over distance and the fading of signals along multiple paths, are the the top three issues we have to address in wireless channels. Interference is an example of **negative externality** that we will encounter many times in this book, together with ways to "internalize" it by designing the right mechanism.

Here is an example of significant interference. As shown in Figure 1.1, a user standing right next to the BS can easily overwhelm another user far away at the edge of the cell. This is the classic **near-far problem** in CDMA networks. It was solved in the IS-95 standard by Qualcomm in 1989. This solution has been one of the cornerstones in realizing the potential of CDMA since then.

Qualcomm's solution to the near-far problem is simple and effective. The receiver infers the channel quality and sends that back to the transmitter as feedback. Consider an uplink transmission: multiple MSs trying to send signals to the BS in a particular cell. The BS can estimate the channel quality from each MS to itself, e.g., by looking at the ratio of the received signal power to the transmitted power, the latter being pre-configured to some value during the channel-estimation timeslot. Then, the BS inverts the channel quality and sends that value, on some feedback control channel, back to the MSs, telling them that these are the gain parameters they should use in setting their transmit powers. In this way, all the received signal strengths will be made equal. This is the basic MS **transmit power control** algorithm in CDMA.

But what if equalization of the received signal powers is *not* the right goal? For voice calls, the typical application on cell phones in 2G networks in the 1990s, there is often a target value of the received signal quality that each call needs to achieve. This signal quality factor is called the Signal to Interference Ratio (**SIR**). It is the ratio between the received signal strength and the sum strength of all the interference (plus the receiver noise strength). Of course, it is easy to raise the SIR for just one user: just increase its transmitter's power. But that translates into higher interference for everyone else, which further leads to higher transmit powers being used by them if they also want to maintain or improve their SIRs. This positive feedback escalates into a transmit power "arms race" until each user is transmitting at the maximum power. That would not be a desirable state to operate in.

If each user fixes a reasonable target SIR, can we do better than this "arms race" through a more intelligent power control? Here, "being reasonable" means that the SIRs targeted by all the users in a cell are mutually compatible; they *can* be simultaneously achieved by some configuration of transmit powers at the MSs.

The answer is yes. In 1992-1993, a sequence of research results developed the basic version of **Distributed Power Control** (DPC), a fully **distributed algorithm**. We will discuss later what we mean by "distributed" and "fully distributed." For now, it suffices to say that, in DPC, each pair of transmitter (e.g., an MS) and receiver (e.g., the BS) does not need to know the transmit power or channel quality of any other pair. At each timeslot, all it needs to know is the actual SIR it currently achieves at the receiver. Then, by taking the ratio between the fixed, target SIR and the variable, actual SIR value measured for this timeslot, and multiplying the current transmit power by that ratio, we get the transmit power for the next timeslot. This update happens simultaneously at each pair of transmitter and receiver.

This simple method is an **iterative algorithm**; the updates continue from one timeslot to the next, unlike with the one-shot, received-power-equalization algorithm. But it is still simple, and when the target SIRs can be simultaneously achieved, it has been proven to **converge**: the iterative procedure will stop over time. When it stops, it stops at the right solution: a power-minimal configuration of transmit powers that achieves the target SIRs for all. DPC converges quite fast, approaching the right power levels with an error that decays as a geometric series. DPC can even be carried out asynchronously: each radio has a different clock and therefore different definitions of what timeslot it is now.

Of course, in real systems the timeslots are indeed *asynchronous* and power levels are *discrete*. Asynchronous and quantized versions of DPC have been implemented in all the CDMA standards in 3G networks. Some standards run power control 1500 times every second, while others run 800 times a second. Some discretize power levels to 0.1 dB, while others between 0.2 and 0.5 dB. Without CDMA, our cellular networks today could not work as efficiently. Without power control algorithms (and the associated handoff method to support user mobility), CDMA could not function properly. In Chapter 19, we will discuss a 4G standard called LTE. It uses a technology called OFDM instead of CDMA, but power control is still employed for interference reduction and for energy management.

Later, in Chapter 18, we will discuss some of the latest ideas that help further push the data rates in new wireless network standards, ranging from splitting, shrinking, and adjusting the cells to overlaying small cells on top of large ones for traffic offloading, and from leveraging multiple antennas and tilting their positions to "chopping up" the frequency bands for more efficient signal processing.

1.2 A Long Answer

1.2.1 Distributed power control

Before we proceed to a general discussion of the Distributed Power Control (DPC) algorithm, we must first define some symbols.

Consider N pairs of transmitters and receivers. Each pair forms a (logical) link, indexed by i. The transmit power of the transmitter of link i is p_i, some positive number, usually capped at a maximum value: $p_i \leq p_{max}$ (although we will not consider the effect of this cap in the analysis of the algorithm). The transmitted power impacts both the received power at the intended receiver and the received interference at the receivers of all other pairs.

Now, consider the channel from the transmitter of link (i.e., transmitter–receiver pair) j to the receiver of link i, and denote the **channel gain** by G_{ij}. So G_{ii} is the direct channel gain; the bigger the better, since it is the channel for the intended transmission for the transmitter–receiver pair of link i. All the other $\{G_{ij}\}$, for j not equal to i, are gains for interference channels, so the smaller the better. We call these channel "gains", but actually they are less than 1, so maybe a better term is channel "loss."

This notation is visualized in Figure 1.3 for a simple case of two MSs talking to a BS, which can be thought of as two different (logically separated) receivers physically located together.

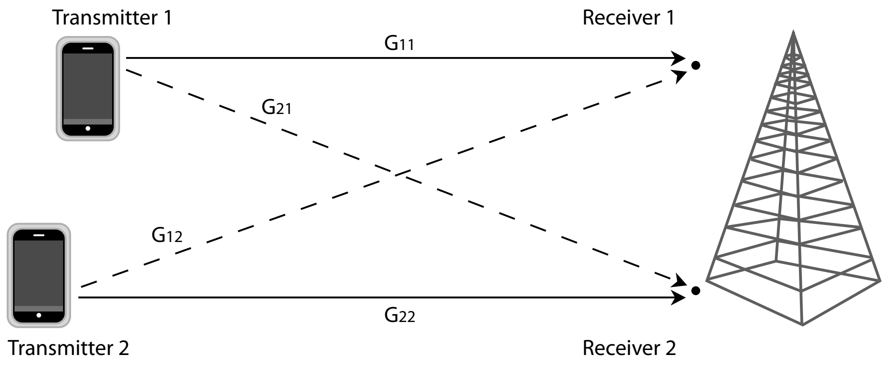

Figure 1.3 Uplink interference between two mobile stations at the base station. We can think of the base station as two (logically separated) receivers collocated. G_{11} and G_{22} are direct channel gains, the bigger the better. G_{12} and G_{21} are interference channel gains, the smaller the better.

Each G_{ij} is determined by two main factors: (1) location of the transmitter and receiver and (2) the quality of the channel in between. G_{ii} is also enhanced by the CDMA spreading codes that help the intended receivers decode more accurately.

The received power of the intended transmission at the receiver is therefore $G_{ii}p_i$. What about the interference? It is the sum of $G_{ij}p_j$ over all transmitters j (other than the intended one i): $\sum_{j \neq i} G_{ij}p_j$. There is also noise n_i in the receiver electronics for each receiver i. So we can write the SIR, a unit-less ratio, at the receiver of logical link i as

$$\text{SIR}_i = \frac{G_{ii}p_i}{\sum_{j \neq i} G_{ij}p_j + n_i}. \tag{1.1}$$

For proper decoding of the packets, the receiver needs to maintain a target level of SIR. We will denote that as γ_i for link i, and we want $\text{SIR}_i \geq \gamma_i$ for all i. Clearly, increasing p_1 raises the SIR for receiver 1 but lowers the SIR for all other receivers.

As in a typical algorithm we will encounter throughout this book, we assume that time is divided into discrete slots, each indexed by $[t]$. At each timeslot t, the receiver on link i can measure the received SIR readily, and feeds back that number, $\text{SIR}_i[t]$, to the transmitter.

The DPC algorithm can be described through a simple equation: each transmitter simply multiplies the current power level $p_i[t]$ by the ratio between the target SIR, γ_i, and the current measured $\text{SIR}_i[t]$, to obtain the power level to use in the next timeslot:

$$p_i[t+1] = \frac{\gamma_i}{\text{SIR}_i[t]} p_i[t], \quad \text{for each } i. \tag{1.2}$$

We will use the symbols $\forall i$ later to say "for each i."

We see that each receiver i needs to measure only its own SIR at each iteration, and each transmitter only needs to remember its own target SIR. There is no need for passing any control message around, like telling other users what power level it is using. Simple in *communication*, it is a *very* distributed algorithm, and we will later encounter many types of distributed algorithms in various kinds of networks.

This algorithm is also simple in its *computation*: just one division and one multiplication. And it is simple in its parameter *configuration*: there are actually no parameters in the algorithm that need to be tuned, unlike in quite a few other algorithms in later chapters. Simplicity in communication, computation, and configuration is a key reason why certain algorithms are widely adopted in practice.

Intuitively, this algorithm makes sense. First, when the iterations stop because no one's power is changing any more, i.e., when we have convergence to an **equilibrium**, we can see that $\text{SIR}_i = \gamma_i$ for all i.

Second, there is hope that the algorithm will actually converge, given the direction in which the power levels are moving. The transmit power moves up when the received SIR is below the target, and moves down when it is above the target. *Proving* that convergence will happen is not as easy. As one transmitter changes its power, the other transmitters are doing the same, and it is unclear what the next timeslot's SIR values will be. In fact, this algorithm does *not* converge if too many γ_i are too large, i.e., when too many users request large SIRs as their targets.

Third, if satisfying the target SIRs is the only criterion, there are many transmit power configurations that can do that. If $p_1 = p_2 = 1$ mW achieves these two users' target SIRs, $p_1 = p_2 = 10$ mW will do so too. We would like to pick the configuration that uses the least amount of power; we want a power-minimal solution. And the algorithm above seems to be lowering the power when a high power is unnecessary.

DPC illustrates a recurring theme in this book. We will see in almost every chapter that individual's behaviors driven by self-interest can often aggregate into a fair and efficient state globally across all users, especially when there are proper feedback signals. In contrast, either a centralized control or purely random individual actions would have imposed significant downsides.

1.2.2 DPC as an optimization solution

In general, "will it converge?" and "will it converge to the right solution?" are the top two questions that we would like to address in the design of all iterative algorithms. Of course, what "the right solution" means will depend on the definition of optimality. In this case, power-minimal transmit powers that achieve the target SIRs for all users are the "right solution." Power minimization is the **objective** and achieving target SIRs for all users is the **constraint**.

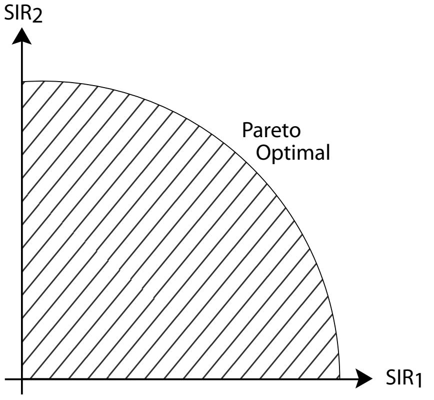

Figure 1.4 An illustration of the SIR feasibility region. It is a constraint set for power control optimization, and visualizes the competition among users. Every point strictly inside the shaded region is a feasible vector of target SIRs. Every point outside is infeasible. And every point on the boundary of the curve is Pareto optimal: you cannot increase one user's SIR without reducing another user's SIR.

In this case, there are many ways to address these questions, for example, using machineries from optimization theory or game theory. Either way, we can show that, under the condition that the target SIR values are indeed **achievable** at all (i.e., there are some values of the transmit powers that can achieve the target SIRs for all users), DPC will converge, and converge to the right solution.

We can illustrate a typical set of feasible SIRs in the SIR feasibility region shown in Figure 1.4. Clearly, we want to operate on the boundary of this region, and each point on the boundary is called **Pareto optimal**. Along this boundary, one user's higher SIR can be achieved, but at the expense of a lower SIR for another user. This highlights another recurrent theme in this book: the need to tackle tradeoffs among competing users and across different design objectives, and the importance of providing incentives for people to react to. There is no free lunch; and we have to balance the benefits with the costs.

It turns out that DPC also solves a global optimization problem for the network. Here, "global" means that the interests of *all* the users are incorporated. In this case, it is the sum of transmit powers that is minimized, and every user's target SIR must be met.

Once we have an objective function and a set of constraints, and we have defined which quantities are variables and which are constants, an **optimization** problem is formulated. In this case, the transmit powers are the variables. The achieved SIRs are also variables, but are derived from the powers. All the other quantities are the constants: they are not degrees of freedom under your control.

If the variable vector $\mathbf{x}_0$ satisfies all the constraints, it is called a **feasible solution**. If an optimization problem's constraints are not mutually compatible, it is called **infeasible**. If an $\mathbf{x}^*$ is both feasible and better than any other feasible solution, i.e., gives the smallest objective function value for a minimization problem (or the largest objective function value for a maximization problem), it is called an **optimal** solution. An optimal solution might not exist, e.g., minimize $1/x$ for $x \in \mathcal{R}$. And optimal solutions may not be unique either.

Here is the optimization problem of varying transmit power to satisfy fixed target SIR constraints and then minimize the total power:

$$\begin{array}{ll} \text{minimize} & \sum_i p_i \\ \text{subject to} & \mathsf{SIR}_i(\mathbf{p}) \geq \gamma_i, \ \ \forall i \\ \text{variables} & \mathbf{p}. \end{array} \tag{1.3}$$

Problem (1.3) looks complicated if we substitute the definition (1.1) of the SIR as a function of the whole vector $\mathbf{p}$:

$$\begin{array}{ll} \text{minimize} & \sum_i p_i \\ \text{subject to} & \frac{G_{ii}p_i}{\sum_{j\neq i} G_{ij}p_j + n_i} \geq \gamma_i, \ \ \forall i \\ \text{variables} & \mathbf{p}. \end{array}$$

But it can be simplified through a different representation. We can easily rewrite it as a **linear programming** problem: minimizing a linear function of the variables subject to linear constraints of the variables:

$$\begin{array}{ll} \text{minimize} & \sum_i p_i \\ \text{subject to} & G_{ii}p_i - \gamma_i(\sum_{j\neq i} G_{ij}p_j + n_i) \geq 0, \ \ \forall i \\ \text{variables} & \mathbf{p}. \end{array}$$

Linear programming problems are easy optimization problems. More generally, convex optimization (to be introduced in Chapter 4) is easy; easy in theory in terms of complexity and easy in practice with fast solution software. In the Advanced Material, we will derive DPC as the solution to problem (1.3).

We will be formulating and solving optimization problems many times in future chapters. Generally, solving a global optimization via local actions by each user requires explicit signaling. But in this case, it turns out that the selfish behavior of users in their own power minimization also solves the global, constrained optimization problem; their interests are correctly aligned already. This is more of an exception than the norm.

1.2.3 DPC as a game

Power control is a competition. One user's received power is another's interference. Each player searches for the right "move" (or, in this case, the right transmit power) so that its "payoff" is optimized (in this case, the transmit power is the smallest possible while providing the user with its target SIR γ_i). We also hope that the whole network reaches some desirable equilibrium as each player strategizes. The concepts of "players," "move," "payoff," and "equilibrium" can be defined in a precise and useful way.

We can model *competition* as a **game**. The word “game” here carries a technical meaning. The study of games is a branch of mathematics called game theory. If the competition is among human beings, a game might actually correspond to people’s strategies. If it is among devices, as in this case among radios, a game is more like an angle of interpretation and a tool for analysis. It turns out that *cooperation* can also be modeled in the language of game theory, as we will show in Chapter 6.

In the formal definition, a game is specified by three elements:

1. a set of **players** $\{1, 2, \ldots, N\}$,
2. a **strategy space** A_i for each player i, and
3. a **payoff function**, or utility function, U_i for each player to maximize (or a **cost function** to minimize). Function U_i maps each combination of all players’ strategies to a real number, the payoff (or cost), to player i.

	`Not Confess`	`Confess`
`Not Confess`	$(-1, -1)$	$(-5, 0)$
`Confess`	$(0, -5)$	$(-3, -3)$

Table 1.1 Prisoner’s dilemma. This is a famous game in which there is a unique and undesirable Nash equilibrium. Player A’s two strategies are the two rows. Player B’s two strategies are the two columns. The values in the table represent the payoffs to the two players in each scenario.

Now consider the two-player game in Table 1.1. This is the famous **prisoner’s dilemma** game, which we will also encounter later in voting paradoxes, tragedy of the commons, and P2P file sharing. The two players are two prisoners. Player A’s strategies are shown in rows and player B’s in columns. Each entry in the 2×2 table has two numbers, (x, y), where x is the payoff to A and y that to B if the two players pick the corresponding strategies. As you would expect from the coupling between the players, each payoff value is determined jointly by the strategies of both players. For example, the payoff function maps `(Not Confess, Not Confess)` to -1 for both players A and B. These payoffs are negative because they are the numbers of years the two prisoners are going to serve in prison. If one confesses but the other does not, the one who confesses gets a deal to walk away free and the other one is heavily penalized. If both confess, both serve three years. If neither confesses, only a lesser conviction can be pursued and both serve only one year. Both players know this table, but they cannot communicate with each other.

If player A chooses the strategy `Not Confess`, player B should choose the strategy `Confess`, since $0 > -1$. This is called the **best response strategy** by player B, in response to player A choosing the strategy `Not Confess`.

If player A chooses the strategy `Confess`, player B’s best response strategy is still `Confess`, since $-3 > -5$. When the best response strategy of a player is the *same* no matter what strategy the other player chooses, we call that a **dominant strategy**. It might not exist. But, when it does, a player will obviously pick a dominant strategy.

In this case, `Confess` is the dominant strategy for player B. By symmetry, it is also the dominant strategy for player A. So both players will pick `Confess`, and `(Confess, Confess)` is an **equilibrium** for the game. This is a slightly different definition of "equilibrium" from what we saw before, where equilibrium means an update equation reaches a fixed point.

Clearly, this equilibrium is undesirable: `(Not Confess, Not Confess)` gives a higher payoff value to both players: -1 instead of -3. But the two prisoners could not have coordinated to achieve `(Not Confess, Not Confess)`. An equilibrium might not be **socially optimal**, i.e., a set of strategies maximizing the sum of payoffs $\sum_i U_i$ of all the players. It might not even be Pareto optimal, i.e., a set of strategies such that no player's payoff can be increased without hurting another player's payoff.

	`Action Movie`	`Romance Movie`
`Action Movie`	(2, 1)	(0, 0)
`Romance Movie`	(0, 0)	(1, 2)

Table 1.2 Coordination Game. In this game, there are two Nash equilibria. Neither player has an incentive to unilaterally change strategy in either equilibrium. Rows are your strategies and columns your friend's.

Consider a different game in Table 1.2. This is a typical game model for **coordination**, a task we will see many times in future chapters. You and your friend are trying to coordinate which movie to watch together. If you disagree, you will not go to any movie and the payoff is zero for both of you. If you agree, each will get some positive payoff but the values are different, as you prefer the action movie and your friend prefers the romance movie. (By the way, we will try to understand how to predict a person's preference for different types of movies in Chapter 4.)

In this game, there is no dominant strategy, but it so happens that there are pairs of best response strategies that match each other. If my best response to my friend picking strategy a is strategy b, and my friend's best response to my picking strategy b is strategy a, then the (a, b) pair "matches."

In the coordination game, `(Action, Action)` is such a pair, and `(Romance, Romance)` is another pair. For both pairs, neither player has an incentive to *unilaterally* move away from its choice in this pair of strategies. If both move at the same time, they could both benefit, but neither wants to do that *alone*. This creates an equilibrium in strategic thinking: I will not move unless you move, and you think the same way too. This is called a **Nash equilibrium**. In the prisoner's dilemma, `(Confess, Confess)` is a Nash equilibrium. In the coordination game, both `(Action, Action)` and `(Romance, Romance)` are Nash equilibria.

Symbolically, for a two-user game, suppose the two payoff functions are (U_1, U_2) and the two strategy spaces are $(\mathcal{A}, \mathcal{B})$ for the two players, respectively. We say $(a^* \in \mathcal{A}, b^* \in \mathcal{B})$ is a Nash equilibrium if

$$U_1(a^*, b^*) \geq U_1(a, b^*), \quad \text{for any } a \in \mathcal{A},$$

and

$$U_2(a^*, b^*) \geq U_2(a^*, b), \quad \text{for any } b \in \mathcal{B}.$$

A Nash equilibrium might not exist in a game. And when it exists, it need not be unique (like in the coordination game above), or socially optimal, or Pareto optimal. But if the players are allowed to throw a coin and decide *probabilistically* which strategy to play, i.e., a **mixed strategy**, it is guaranteed, by Nash's famous result, that a Nash equilibrium always exists.

We will expand and use our game-theoretic language in several future chapters. In our current case of power control as a game, the set of players is the set of transmitters. The strategy for each player is its transmit power level, and the strategy space is the set of transmit powers so that the target SIR is achieved. The cost function to minimize is the power level itself.

In this power control game, while the cost function is independent across the players, each player's strategy space A_i depends on the transmit powers of all the other players. This coupling across the players is introduced by the very nature of interference, and leads to strategic moves by the players.

Here is a simple fact, which is important to realize and easy to verify: iterating by the best response strategy of each player in the power control game is given precisely by (1.2). Given that all the other transmitters transmit at a certain power level, my best response strategy is to pick the power level that is my current power level times the ratio between my target and the current SIRs.

Now, consider player 1. If the transmit powers of all the other players become smaller, player 1's strategy space A_1 will be larger. (This is just another way to say there is negative externality.) There are more transmit powers to pick while still being able to maintain the target SIR, since other players' transmit powers are smaller and the denominator in the SIR is smaller. Given this sense of monotonicity, we suspect that maybe best responses converge. Indeed it can be shown that, for games with this property (and under some technicalities), the sequence of best response strategies converge.

This game-theoretic approach is one of the ways to prove the convergence of DPC. Another way is through the angle of global optimization and with the help of linear algebra, as we will present in the Advanced Material. But first, a small and detailed example.

1.3 Examples

A word about all the examples in this book. Most of them are completely numerical and presented in great detail, so as to alleviate any "symbol-phobia" a reader may have. This means that we have to strike a tradeoff between a realistic size for illustration and a small size to fit in a few pages. In some cases, these examples do not represent the actual scale of the problems, while scaling-up can actually be a core difficulty.

Suppose we have four (transmitter, receiver) pairs. Let the channel gains $\{G_{ij}\}$ be given in Table 1.3. As the table suggests, we can also represent these gains in a matrix. You can see that, in general, $G_{ij} \neq G_{ji}$ because the interference channels do not have to be symmetric.

Receiver	Transmitter of Link			
of Link	1	2	3	4
1	1	0.1	0.2	0.3
2	0.2	1	0.1	0.1
3	0.2	0.1	1	0.1
4	0.1	0.1	0.1	1

Table 1.3 Channel gains in an example of DPC. The entries are for illustrating the algorithm. They do not represent actual numerical values typically observed in real cellular networks.

Suppose that the initial power level is 1.0 mW on each link, and that the noise on each link is 0.1 mW. Then, the initial signal-to-interference ratios are given by

$$\begin{aligned}
\mathrm{SIR}_1[0] &= \tfrac{1\times1.0}{0.1\times1.0+0.2\times1.0+0.3\times1.0+0.1} = 1.43,\\
\mathrm{SIR}_2[0] &= \tfrac{1\times1.0}{0.2\times1.0+0.1\times1.0+0.1\times1.0+0.1} = 2.00,\\
\mathrm{SIR}_3[0] &= \tfrac{1\times1.0}{0.2\times1.0+0.1\times1.0+0.1\times1.0+0.1} = 2.00,\\
\mathrm{SIR}_4[0] &= \tfrac{1\times1.0}{0.1\times1.0+0.1\times1.0+0.1\times1.0+0.1} = 2.50,
\end{aligned}$$

where we use the formula

$$\mathrm{SIR}_i = \frac{G_{ii}p_i}{\sum\limits_{j\neq i} G_{ij}p_j + n_i},$$

with p_i representing the power level of link i and n_i the noise on link i. These SIR values are shown on a linear scale rather than the log (dB) scale in this example.

We will use DPC to adjust the power levels. Suppose that the target SIRs are

$$\begin{aligned}
\gamma_1 &= 2.0,\\
\gamma_2 &= 2.5,\\
\gamma_3 &= 1.5,\\
\gamma_4 &= 2.0.
\end{aligned}$$

Then the new power levels are, in mW,

$$\begin{aligned}
p_1[1] &= \tfrac{\gamma_1}{\mathrm{SIR}_1[0]} p_1[0] = \tfrac{2.0}{1.43} \times 1.0 = 1.40,\\
p_2[1] &= \tfrac{\gamma_2}{\mathrm{SIR}_2[0]} p_2[0] = \tfrac{2.5}{2.00} \times 1.0 = 1.25,\\
p_3[1] &= \tfrac{\gamma_3}{\mathrm{SIR}_3[0]} p_3[0] = \tfrac{1.5}{2.00} \times 1.0 = 0.75,\\
p_4[1] &= \tfrac{\gamma_4}{\mathrm{SIR}_4[0]} p_4[0] = \tfrac{2.0}{2.5} \times 1.0 = 0.80.
\end{aligned}$$

Now each receiver calculates the new SIR and feeds it back to its transmitter:

$$\begin{aligned}
\mathrm{SIR}_1[1] &= \tfrac{1\times1.40}{0.1\times1.25+0.2\times0.75+0.3\times0.8+0.1} = 2.28,\\
\mathrm{SIR}_2[1] &= \tfrac{1\times1.25}{0.2\times1.40+0.1\times0.75+0.1\times0.8+0.1} = 2.34,\\
\mathrm{SIR}_3[1] &= \tfrac{1\times0.75}{0.2\times1.40+0.1\times1.25+0.1\times0.8+0.1} = 1.28,\\
\mathrm{SIR}_4[1] &= \tfrac{1\times0.80}{0.1\times1.40+0.1\times1.25+0.1\times0.75+0.1} = 1.82.
\end{aligned}$$

The new power levels in the next timeslot become, in mW,

$$\begin{aligned}
p_1[2] &= \tfrac{\gamma_1}{\mathrm{SIR}_1[1]} p_1[1] = \tfrac{2.0}{2.28} \times 1.40 = 1.23,\\
p_2[2] &= \tfrac{\gamma_2}{\mathrm{SIR}_2[1]} p_2[1] = \tfrac{2.5}{2.33} \times 1.25 = 1.34,\\
p_3[2] &= \tfrac{\gamma_3}{\mathrm{SIR}_3[1]} p_3[1] = \tfrac{1.5}{1.28} \times 0.75 = 0.88,\\
p_4[2] &= \tfrac{\gamma_4}{\mathrm{SIR}_4[1]} p_4[1] = \tfrac{2.0}{1.82} \times 0.80 = 0.88,
\end{aligned}$$

with the corresponding SIRs as follows:

$$\begin{aligned}
\mathrm{SIR}_1[2] &= \tfrac{1\times1.23}{0.1\times1.34+0.2\times0.88+0.3\times0.88+0.1} = 1.83,\\
\mathrm{SIR}_2[2] &= \tfrac{1\times1.34}{0.2\times1.23+0.1\times0.88+0.1\times0.88+0.1} = 2.56,\\
\mathrm{SIR}_3[2] &= \tfrac{1\times0.88}{0.2\times1.23+0.1\times1.34+0.1\times0.88+0.1} = 1.55,\\
\mathrm{SIR}_4[2] &= \tfrac{1\times0.88}{0.1\times1.23+0.1\times1.34+0.1\times0.88+0.1} = 1.98.
\end{aligned}$$

Calculating the new power levels again, we have, in mW,

$$\begin{aligned}
p_1[3] &= \tfrac{\gamma_1}{\mathrm{SIR}_1[2]} p_1[2] = \tfrac{2.0}{1.83} \times 1.23 = 1.35,\\
p_2[3] &= \tfrac{\gamma_2}{\mathrm{SIR}_2[2]} p_2[2] = \tfrac{2.5}{2.56} \times 1.34 = 1.30,\\
p_3[3] &= \tfrac{\gamma_3}{\mathrm{SIR}_3[2]} p_3[2] = \tfrac{1.5}{1.55} \times 0.88 = 0.85,\\
p_4[3] &= \tfrac{\gamma_4}{\mathrm{SIR}_4[2]} p_4[2] = \tfrac{2.0}{1.98} \times 0.88 = 0.89.
\end{aligned}$$

Then the new SIRs are

$$\begin{aligned}
\mathrm{SIR}_1[3] &= \tfrac{1\times1.35}{0.1\times1.30+0.2\times0.85+0.3\times0.89+0.1} = 2.02,\\
\mathrm{SIR}_2[3] &= \tfrac{1\times1.30}{0.2\times1.35+0.1\times0.85+0.1\times0.89+0.1} = 2.40,\\
\mathrm{SIR}_3[3] &= \tfrac{1\times0.85}{0.2\times1.35+0.1\times1.30+0.1\times0.89+0.1} = 1.45,\\
\mathrm{SIR}_4[3] &= \tfrac{1\times0.89}{0.1\times1.35+0.1\times1.30+0.1\times0.85+0.1} = 1.97,
\end{aligned}$$

and the new power levels, in mW, are

$$\begin{aligned}
p_1[4] &= \tfrac{\gamma_1}{\mathrm{SIR}_1[3]}p_1[3] = \tfrac{2.0}{2.02} \times 1.35 = 1.33,\\
p_2[4] &= \tfrac{\gamma_2}{\mathrm{SIR}_2[3]}p_2[3] = \tfrac{2.5}{2.40} \times 1.30 = 1.36,\\
p_3[4] &= \tfrac{\gamma_3}{\mathrm{SIR}_3[3}p_3[3] = \tfrac{1.5}{1.45} \times 0.85 = 0.88,\\
p_4[4] &= \tfrac{\gamma_4}{\mathrm{SIR}_4[3]}p_4[3] = \tfrac{2.0}{1.97} \times 0.89 = 0.90.
\end{aligned}$$

We see that the power levels are beginning to converge: p_1, p_2, p_3 and p_4 all change by less than 0.1 mW now. The new SIRs are

$$\begin{aligned}
\mathrm{SIR}_1[4] &= \tfrac{1\times1.33}{0.1\times1.36+0.2\times0.88+0.3\times0.90+0.1} = 1.96,\\
\mathrm{SIR}_2[4] &= \tfrac{1\times1.36}{0.2\times1.33+0.1\times0.88+0.1\times0.90+0.1} = 2.49,\\
\mathrm{SIR}_3[4] &= \tfrac{1\times0.88}{0.2\times1.33+0.1\times1.36+0.1\times0.90+0.1} = 1.49,\\
\mathrm{SIR}_4[4] &= \tfrac{1\times0.90}{0.1\times1.33+0.1\times1.36+0.1\times0.88+0.1} = 1.97.
\end{aligned}$$

Iterating one more time, the new power levels, in mW, are

$$\begin{aligned}
p_1[5] &= \tfrac{\gamma_1}{\mathrm{SIR}_1[4]}p_1[4] = \tfrac{2.0}{1.96} \times 1.33 = 1.37,\\
p_2[5] &= \tfrac{\gamma_2}{\mathrm{SIR}_2[4]}p_2[4] = \tfrac{2.5}{2.49} \times 1.36 = 1.36,\\
p_3[5] &= \tfrac{\gamma_3}{\mathrm{SIR}_3[4]}p_3[4] = \tfrac{1.5}{1.49} \times 0.88 = 0.89,\\
p_4[5] &= \tfrac{\gamma_4}{\mathrm{SIR}_4[4]}p_4[4] = \tfrac{2.0}{1.97} \times 0.90 = 0.92,
\end{aligned}$$

with corresponding SIRs:

$$\begin{aligned}
\mathrm{SIR}_1[5] &= \tfrac{1\times1.37}{0.1\times1.36+0.2\times0.89+0.3\times0.92+0.1} = 1.98,\\
\mathrm{SIR}_2[5] &= \tfrac{1\times1.36}{0.2\times1.37+0.1\times0.89+0.1\times0.92+0.1} = 2.45,\\
\mathrm{SIR}_3[5] &= \tfrac{1\times0.89}{0.2\times1.37+0.1\times1.36+0.1\times0.92+0.1} = 1.48,\\
\mathrm{SIR}_4[5] &= \tfrac{1\times0.92}{0.1\times1.37+0.1\times1.36+0.1\times0.89+0.1} = 1.98.
\end{aligned}$$

All the SIRs are now within 0.05 of the target. The power levels keep iterating, taking the SIRs closer to the target. Figure 1.5 shows the graph of power level versus the number of iterations. After about 20 iterations, the change is too small to be seen on the graph; the power levels at that time are

$$\begin{aligned}
p_1 &= 1.46 \text{ mW},\\
p_2 &= 1.46 \text{ mW},\\
p_3 &= 0.95 \text{ mW},\\
p_4 &= 0.97 \text{ mW}.
\end{aligned}$$

The resulting SIRs are shown in Figure 1.6. We get very close to the target SIRs, by visual inspection, after about 10 iterations.

While we are at this example, let us also walk through a compact, matrix representation of the target SIR constraints. This will be useful in the next section.

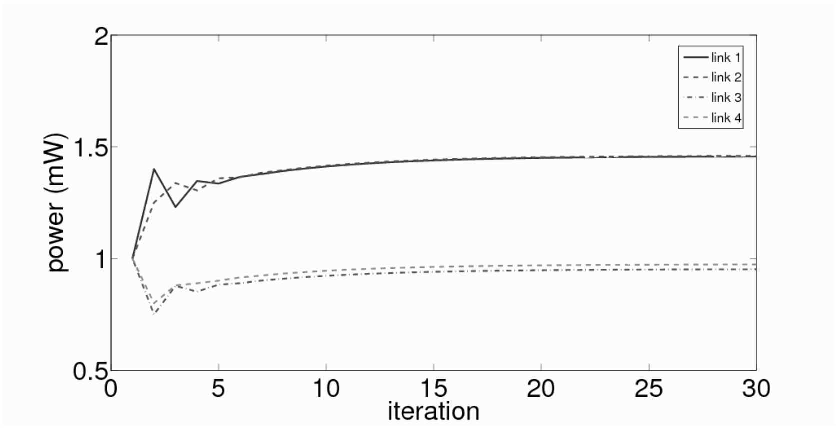

Figure 1.5 The convergence of power levels in an example of DPC.

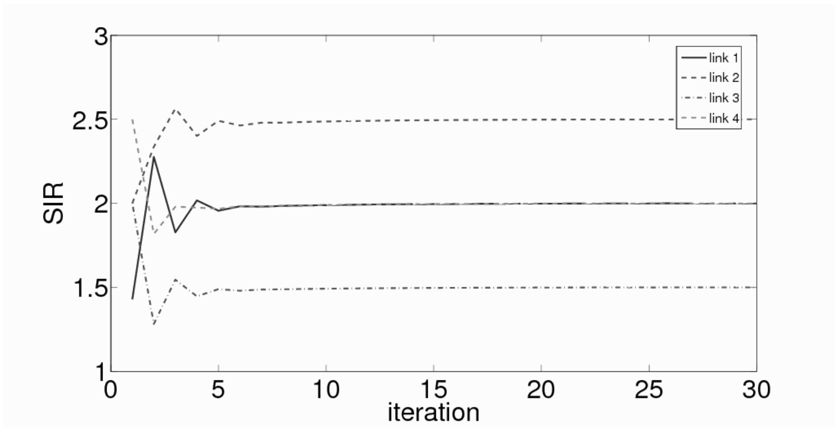

Figure 1.6 The convergence of SIRs in an example of DPC.

If the target SIR γ_i is achieved or exceeded by $\{p_i\}$, we have

$$\frac{G_{ii}p_i}{\displaystyle\sum_{j\neq i} G_{ij}p_j + n_i} \geq \gamma_i,$$

for $i = 1, 2, 3, 4$. On multiplying both sides by $\sum_{j\neq i} G_{ij}p_j + n_i$ and dividing by G_{ii}, we have

$$p_i \geq \frac{\gamma_i}{G_{ii}} \left(\sum_{j\neq i} G_{ij}p_j + n_i \right),$$

which can be written as

$$p_i \geq \gamma_i \sum_{j\neq i} \frac{G_{ij}}{G_{ii}} p_j + \frac{\gamma_i}{G_{ii}} n_i. \tag{1.4}$$

Now we define the variable vector

$$\mathbf{p} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix},$$

and a constant vector

$$\mathbf{v} = \begin{bmatrix} \frac{\gamma_1 n_1}{G_{11}} \\ \frac{\gamma_2 n_2}{G_{22}} \\ \frac{\gamma_3 n_3}{G_{33}} \\ \frac{\gamma_4 n_4}{G_{44}} \end{bmatrix} = \begin{bmatrix} \frac{2.0\times 0.1}{1.0} \\ \frac{2.5\times 0.1}{1.0} \\ \frac{1.5\times 0.1}{1.0} \\ \frac{2.0\times 0.1}{1.0} \end{bmatrix} = \begin{bmatrix} 0.20 \\ 0.25 \\ 0.15 \\ 0.20 \end{bmatrix}.$$

Define also a 4×4 diagonal matrix $\mathbf{D}$ with γ_i on the diagonal, and another 4×4 matrix $\mathbf{F}$ where $F_{ij} = G_{ij}/G_{ii}$ for $i \neq j$, and the diagonal entries of F are zero. Plugging in the numbers, we have

$$\mathbf{D} = \begin{bmatrix} 2.0 & 0 & 0 & 0 \\ 0 & 2.5 & 0 & 0 \\ 0 & 0 & 1.5 & 0 \\ 0 & 0 & 0 & 2.0 \end{bmatrix},$$

$$\mathbf{F} = \begin{bmatrix} 0 & 0.1 & 0.2 & 0.3 \\ 0.2 & 0 & 0.1 & 0.1 \\ 0.2 & 0.1 & 0 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0 \end{bmatrix}.$$

We can now rewrite (1.4) as

$$\mathbf{p} \geq \mathbf{DFp} + \mathbf{v} = \begin{bmatrix} 0 & 0.20 & 0.40 & 0.60 \\ 0.50 & 0 & 0.25 & 0.25 \\ 0.30 & 0.15 & 0 & 0.15 \\ 0.20 & 0.20 & 0.20 & 0 \end{bmatrix} \mathbf{p} + \begin{bmatrix} 0.20 \\ 0.25 \\ 0.15 \\ 0.20 \end{bmatrix},$$

where $\geq$ between two vectors (of equal length) simply represents component-wise inequality between the corresponding entries of the two vectors.

We can check that the power levels in the last iteration shown above satisfy this inequality tightly:

$$\begin{bmatrix} 1.46 \\ 1.46 \\ 0.95 \\ 0.97 \end{bmatrix} \geq \begin{bmatrix} 0 & 0.20 & 0.40 & 0.60 \\ 0.50 & 0 & 0.25 & 0.25 \\ 0.30 & 0.15 & 0 & 0.15 \\ 0.20 & 0.20 & 0.20 & 0 \end{bmatrix} \begin{bmatrix} 1.46 \\ 1.46 \\ 0.95 \\ 0.97 \end{bmatrix} + \begin{bmatrix} 0.20 \\ 0.25 \\ 0.15 \\ 0.20 \end{bmatrix} = \begin{bmatrix} 1.46 \\ 1.46 \\ 0.95 \\ 0.97 \end{bmatrix}.$$

We will use this matrix representation as we go from problem representation (1.3) to problem representation (1.5) in the next section. We will also see in later chapters many different matrices that summarize the *topology* of a network, and operations involving these matrices that model *functionalities* running on the network.

1.4 Advanced Material

1.4.1 Iterative power method

We can generalize the vector notation in the last example. Let $\mathbf{1}$ represent a vector of 1s, so the objective function is simply $\mathbf{1}^T\mathbf{p} = \sum_i p_i$. Let $\mathbf{I}$ be the identity matrix, $\mathbf{D}$ a diagonal matrix with the diagonal entries being the target SIR values $\{\gamma_i\}$, and $\mathbf{F}$ be a matrix capturing the given channel conditions: $F_{ij} = G_{ij}/G_{ii}$ if $i \neq j$, and $F_{ii} = 0$. The constant vector $\mathbf{v}$ captures the normalized noise levels, with $v_i = \gamma_i n_i / G_{ii}$. We will soon see why this shorthand notation is useful.

Equipped with the notation above, we can represent the target SIR constraints in problem (1.3) as

$$\mathbf{p} \geq \mathbf{DFp} + \mathbf{v},$$

and further group all the terms involving the variables $\mathbf{p}$. The linear programming problem we have now becomes

$$\begin{array}{ll} \text{minimize} & \mathbf{1}^T\mathbf{p} \\ \text{subject to} & (\mathbf{I} - \mathbf{DF})\mathbf{p} \geq \mathbf{v} \\ \text{variables} & \mathbf{p}. \end{array} \tag{1.5}$$

You should verify that problem (1.3) and problem (1.5) are indeed equivalent, by using the definitions of the SIR, of matrices $(\mathbf{D}, \mathbf{F})$, and of vector $\mathbf{v}$.

Linear programming problems are conceptually and computationally easy to solve in general. Our special case here has even more structure. This $\mathbf{DF}$ matrix is a **non-negative matrix**, since all entries of the matrix are non-negative numbers. Non-negative matrices are a powerful modeling tool in linear algebra, with applications from economics to ecology. They have been well-studied in matrix analysis through the **Perron–Frobenius theory**.

If the largest eigenvalue of the $\mathbf{DF}$ matrix, denoted as $\rho(\mathbf{DF})$, is less than 1, then the following three statements are true.

- (a) We can guarantee that the set of target SIRs can indeed be achieved simultaneously, which makes sense since $\rho(\mathbf{DF}) < 1$ means that the $\{\gamma_i\}$ in $\mathbf{D}$ are not "too big," relative to the given channel conditions captured in $\mathbf{F}$.
- (b) We can invert the matrix defining the linear constraints in our optimization problem (1.5): solve the problem by computing $(\mathbf{I} - \mathbf{DF})^{-1}\mathbf{v}$. But, of course, there is no easy way to directly run this matrix inversion distributively across the MSs.
- (c) The inversion of the matrix can be expressed as a sum of terms, each term a multiplication of matrix $\mathbf{DF}$ by itself. More precisely: $(\mathbf{I} - \mathbf{DF})^{-1} = \sum_{k=0}^{\infty}(\mathbf{DF})^k$. This is an infinite sum, so we say that the partial sum of K

terms, $\sum_{k=0}^{K}(\mathbf{DF})^k$, will converge as K becomes very large. Furthermore, the tail term in this sum, $(\mathbf{DF})^k$, approaches 0 as k becomes large:

$$\lim_{k\to\infty}(\mathbf{DF})^k = 0.$$

The key insight is that we want to invert a matrix $(\mathbf{I}-\mathbf{DF})$, because that will lead us to a power-minimal solution to achieving all the target SIRs:

$$\mathbf{p}^* = (\mathbf{I} - \mathbf{DF})^{-1}\mathbf{v}$$

is a solution of problem (1.3), i.e., for any solution $\hat{\mathbf{p}}$ satisfying the constraints in problem (1.3), $\mathbf{p}^*$ is better:

$$\mathbf{p}^* \leq \hat{\mathbf{p}}.$$

Now, the matrix-inversion step is not readily implementable in a distributed fashion, as we need in cellular network power control. Fortunately, it can be achieved by applying the following update.

(1) First, $\mathbf{p}^* = (\mathbf{I} - \mathbf{DF})^{-1}\mathbf{v}$ can be represented as a power series, as stated in Statement (c) above:

$$\mathbf{p}^* = \sum_{k=0}^{\infty}(\mathbf{DF})^k\mathbf{v}. \tag{1.6}$$

(2) Then, you can readily check that the following iteration over time gives exactly the above power series (1.6), as time t goes on:

$$\mathbf{p}[t+1] = \mathbf{DF}\mathbf{p}[t] + \mathbf{v}. \tag{1.7}$$

One way to check this is to substitute the above recursive formula of $\mathbf{p}$ (1.7) all the way to $\mathbf{p}[0]$, the initialization of the iteration. Then, at any time t, we have

$$\mathbf{p}[t] = (\mathbf{DF})^t\mathbf{p}[0] + \sum_{k=0}^{t-1}(\mathbf{DF})^k\mathbf{v},$$

which converges, as $t \to \infty$, to

$$\lim_{t\to\infty}\mathbf{p}[t] = 0 + \sum_{k=0}^{\infty}(\mathbf{DF})^k\mathbf{v} = \mathbf{p}^*,$$

since $(\mathbf{DF})^t\mathbf{p}[0]$ approaches 0 as $t \to \infty$. So, we know (1.7) is right. This also shows that it does not matter what the initialization vector $\mathbf{p}[0]$ is. Its effect will be washed away as time goes on.

(3) Finally, rewrite the vector form of update equation (1.7) in scalar form for each transmitter i, and you will see that it is exactly the DPC algorithm (1.2):

$$p_i[t+1] = \frac{\gamma_i}{\mathrm{SIR}_i[t]}p_i[t], \quad \text{for each } i.$$

We just completed a development and convergence analysis of the DPC as the solution of a global optimization problem through the language of linear algebra. In general, for any square matrix $\mathbf{A}$, the following statements are equivalent.

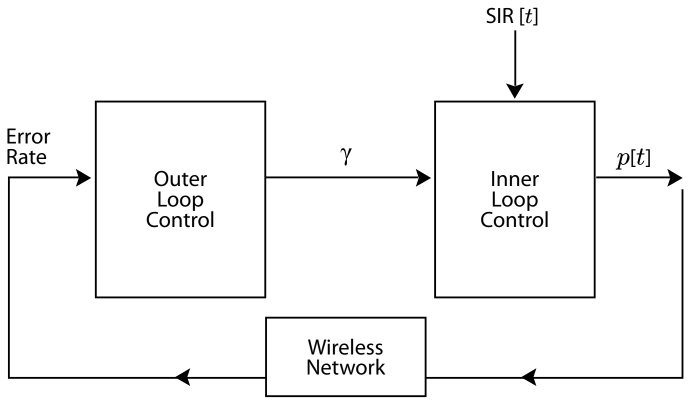

Figure 1.7 Inner and outer loops of power control in cellular networks. The inner loop takes in a fixed target SIR, compares it with the current SIR, and updates the transmit power. The outer loop adjusts the target SIR on the basis of the performance measured over a longer timescale. We have focused on the inner-loop power control in this chapter.

1. The largest eigenvalue is less than 1.
2. The limit of this matrix multiplying itself is 0: $\lim_{k\to\infty} \mathbf{A}^k = 0$.
3. The infinite sum of powers $\sum_{k=0}^{\infty} \mathbf{A}^k$ exists and equals $(\mathbf{I} - \mathbf{A})^{-1}$.

What we saw in DPC is an iterative **power method** (the word "power" here has nothing to do with transmit powers, but instead refers to raising a matrix to some power, i.e., multiplying a matrix by itself many times). It is a common method used to develop iterative algorithms arising from linear systems models. We formulate the problem as a linear program, then we define the solution to the problem as the solution to a linear equation, implement the matrix inversion through a sequence of matrix powers, turn each of those steps into an easy computation at each timeslot, and, finally, achieve the matrix inversion through an iteration in time. This will be used again when we talk about Google's PageRank algorithm in Chapter 3.

1.4.2 Outer loop power control

As shown in the block diagram in Figure 1.7, in cellular networks there are two timescales of power control. What we have been discussing so far is the **inner-loop power control**. Nested outside of that is the **outer-loop power control**, where the target SIRs $\{\gamma_i\}$ are determined.

One standard way to determine target SIRs is to measure the received signal quality, in terms of decoding error probabilities, at the receiver. If the error rate is too high, the target SIR needs to be increased. And if the error rate is lower than necessary, the target SIR can be reduced.

Alternatively, we can also consider optimizing target SIRs as optimization *variables*. This is particularly useful for 3G and 4G networks where data traffic dominates voice traffic in cellular networks. A higher SIR can provide a higher rate at the same signal quality. But every user wants to achieve a higher SIR. So we need to model this either as a network-wide optimization, maximizing the sum of all users' payoff functions, or as a game, with each user maximizing its own payoff function of SIR.

Summary

Box 1 Distributed power control

Different users' signals interfere with each other in the air, leading to a feasible SIR region with a Pareto-optimal boundary. Interference coordination in CDMA networks can be achieved through distributed power control with implicit feedback. It solves an optimization problem for the network in the form of linear programming, and can also be modeled as a non-cooperative game.

Further Reading

This chapter introduces several foundational methodologies: optimization, games, and algorithms, as well as the basics of cellular wireless networks. As a result, there are many interesting texts to read.

1. An early paper quantifying the benefits of CDMA was written by a Qualcomm team, including Andrew Viterbi:
I. M. Jacobs, R. Padovani, A. J. Viterbi, L. A. Weaver, C. E. Wheatley, "On the capacity of a cellular CDMA system," *IEEE Transactions on Vehicular Technology*, vol. 40, no. 2, pp. 303–312, May 1991.

2. The DPC algorithm in the core of this chapter appeared in the following seminal paper:
G. J. Foschini and Z. Miljanic, "A simple distributed autonomous power control algorithm and its convergence," *IEEE Transactions on Vehicular Technology*, vol. 42, no. 3, pp. 641–646, November 1993.

3. Much more discussion on power control algorithms in cellular networks can be found in the following monograph:
M. Chiang, P. Hande, T. Lan, and C. W. Tan, "Power control for cellular wireless networks," *Foundation and Trends in Networking*, vol. 2, no. 4, pp. 381-533, July 2008.

4. A standard reference on linear algebra is the following mathematics textbook, which includes a chapter on non-negative matrices and Perron–Frobenius theory:
R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, 1990.

5. There are many textbooks on game theory, in all types of styles. A comprehensive introduction is
R. B. Myerson, *Game Theory: Analysis of Conflict*, Harvard University Press, 1997.

Problems

1.1 *Distributed power control* ⋆

(a) Consider three pairs of transmitters and receivers in a cell, with the following channel gain matrix $\mathbf{G}$ and noise of 0.1 mW for all the receivers. The target SIRs are also shown below.

$$\mathbf{G} = \begin{bmatrix} 1 & 0.1 & 0.3 \\ 0.2 & 1 & 0.3 \\ 0.2 & 0.2 & 1 \end{bmatrix}, \qquad \gamma = \begin{bmatrix} 1 \\ 1.5 \\ 1 \end{bmatrix}.$$

With an initialization of all transmit powers at 1 mW, run DPC for ten iterations and plot the evolution of transmit powers and received SIRs. You can use any programming language, or even write the steps out by hand.

(b) Now suppose the power levels for logical links 1, 2, and 3 have converged to the equilibrium in (a). A new pair of transmitter and receiver, labeled as logical link 4, shows up in the same cell, with an initial transmit power of 1 mW and demands a target SIR of 1. The new channel gain matrix is shown below.

$$\mathbf{G} = \begin{bmatrix} 1 & 0.1 & 0.3 & 0.1 \\ 0.2 & 1 & 0.3 & 0.1 \\ 0.2 & 0.2 & 1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 1 \end{bmatrix}.$$

Similarly to what you did in (a), show what happens in the next ten timeslots. What happens at the new equilibrium?

1.2 *Power control infeasibility* ⋆⋆

Consider a three-link cell with the link gains G_{ij} shown below. The receivers request $\gamma_1 = 1, \gamma_2 = 2$, and $\gamma_3 = 1$. The noise $n_i = 0.1$ for all i.

$$\mathbf{G} = \begin{bmatrix} 1 & 0.5 & 0.5 \\ 0.5 & 1 & 0.5 \\ 0.5 & 0.5 & 1 \end{bmatrix}.$$

Prove this set of target SIRs is infeasible.

1.3 *A zero-sum game* ⋆

In the following two-user game, the payoffs of users Alice and Bob are exactly negative of each other in all the combinations of strategies (a,a), (a,b), (b,a), (b,b). This models an extreme case of competition, and is called a **zero-sum game**. Is there any pure strategy equilibrium? How many are there?

	a	b
a	(2, −2)	(3, −3)
b	(3, −3)	(4, −4)

1.4 *Mechanism design* ⋆⋆

Consider the game below. There are two players, Alice and Bob, each with two strategies, and the payoffs are shown below. Consider only pure strategy equilibria.

	a	b
a	(0, 2)	(2, 0)
b	(6, 0)	(3, 2)

(a) Is there a Nash equilibrium, and, if so, what is it?

(b) We want to make this game a "better" one. What entries in the table would you change to make the resulting Nash equilibrium unique and socially optimal?

This is an example of **mechanism design**: change the game so as to induce movement of players to a desirable equilibrium. We will see a lot more of mechanism design in future chapters.

1.5 *Repeating prisoner's dilemma* ⋆⋆⋆

(a) Suppose the two prisoners know that they will somehow be caught in the same situation five more times in future years. What will each prisoner's strategy be in choosing between confession and no confession?

(b) Suppose the two prisoners have infinite lifetimes, and there is always a 90% chance that they will be caught in the same situation after each round of this game. What will each prisoner's strategy be now?

2 How does Google sell ad spaces?

2.1 A Short Answer

Much of the web services and online information is “free” today because of the advertisements shown on the websites. It is estimated that the online ad industry worldwide reached $94.2 billion in 2012. Compared with traditional media, online advertisements’ revenue ranked right below TV and above newspapers.

In the early days of the web, i.e., 1994-1995, online advertisements were sold as banners on a per-thousand-impression basis. But seeing an ad does not mean clicking on it or buying the advertised product or service afterwards. In 1997, GoTo (which later became Overture) started selling advertisement spaces on a per-click basis. This middle ground between ad revenue (what the website cares about) and effectiveness of ad (what the advertisers care about) became a commonly accepted foundation for online advertising.

With the rise of Google came one of the most stable online ad market segments: **search ads**, also called *sponsored search*. In 2002, Google started the AdWords service where you can create your ad, attach keywords to it, and send it to Google’s database. When someone searches for a keyword, Google will return a list of search results, as well as a list of ads on the right panel, or even the main panel, if that keyword matches any of the keywords of ads in its database. This process takes place continuously and each advertiser can adjust her bids frequently. There are often many ad auctions happening at the same time too. We will skip these important factors in the basic models in this chapter, focusing just on a single auction.

Now we face three key questions. First, where will your ad appear on the list? We all know that the order of appearance makes a big difference. You will have to pay more to have your ad placed higher in the list. For example, when I did a search for “Banff National Park” on Google in September 2011, I saw an ad for `www.banfflakelouise.com`, a vacation-planning company. This ad was right on top of the main panel, above all the search results on websites and images of Banff National Park. (By the way, how those “real” search results are ordered is the subject of the next chapter.) You also see a list of ads on the right panel, starting with the top one for `www.rockymountaineer.com`, a tourist train company. These two companies probably get most of the clicks, and pay more than the other advertisers for each click. The rest of this chapter delves into the auction methods that allocate these ad spaces according to how much each advertiser is willing to pay.

Second, when will these advertisers pay Google? Only when someone clicks on the link and visits their websites (like I just did, thus contributing to Google's revenue). The average number of times that a viewer of the search result page clicks an ad link, over say one hour, is called the **clickthrough rate**. In a general webpage layout, it may be difficult to rank ad spaces by their positions along a line, but we can always rank them by their clickthrough rates. Let us say the payment by advertisers to Google is *proportional* to the clickthrough rates.

Third, what is in it for the advertisers then? Their revenue derived from placing this particular ad is the product of two things: C, the number of clicks (per unit time, say, one hour), and R, the average revenue (in dollars) generated from each click.

- Let us assume that the number of clicks per hour actually observed is indeed the estimated clickthrough rate, both denoted as C. This is of course not true in general, but it is a reasonable assumption to make for our purpose. We also assume that C is independent of the content in the actual advertisement placed, again a shaky assumption to make the model more tractable.
- As for the average revenue R generated from each click (averaged over all clicks), that highly depends on the nature of the goods or services being advertised and sold. R for each ad-space buyer is assumed to be independent of which ad space she ends up buying, a more reasonable assumption than the one on the independence of C of the advertisement content.

This product, $C \times R$, is the buyer's expected revenue from a particular ad space. It is the **valuation** of the ad space to the buyer. For example, if C is 20 clicks per hour for an ad space and R is \$8 generated per click for an ad-space buyer, the valuation of that space to this buyer is \$160 (per hour). For multiple ad spaces, the valuation of each buyer is a *vector*, with one entry per ad space.

In this discussion, there is one seller, which is Google, many buyers/bidders (the advertisers), and many "goods" (the ad spaces). Each bidder can bid for the ad spaces, and Google will then allocate the ad spaces among the bidders according to some rule, and charge the bidders accordingly.

This process is an **auction**. In general, you can have S sellers, N bidders, and K items in an auction. We will consider only the case with $S = 1$. An ad space auction is a special case of general auctions. Auctions can be analyzed as games, i.e., with a set of players, a strategy set per player, and a payoff function per player.

Depending on the rules of an auction, each bidder may choose to bid in different ways, maybe bidding her true valuation of the ad spaces. It would be nice to design the rules so that such a **truthful bidding** behavior is encouraged. But Google has other considerations too, such as maximizing its total revenue: the sum of the revenue from each bidder, which is in turn the product of the number

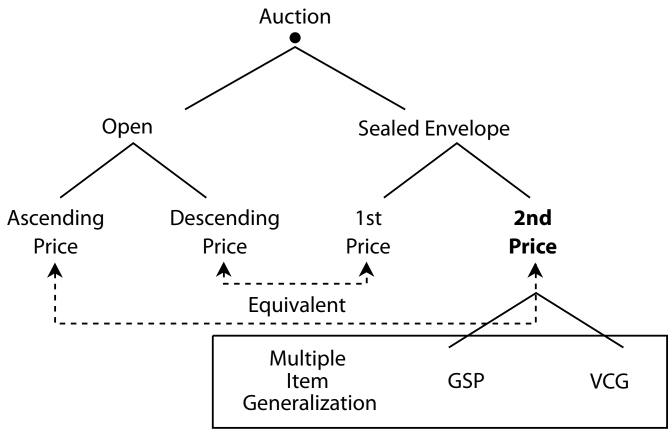

Figure 2.1 A taxonomy of major types of auction in this chapter. The second price, sealed envelope auction is equivalent to (a simpler version of) ascending price open auction, and can be generalized in two different ways to multiple-item auctions: (1) a simple extension to the Generalized Second Price (GSP) auction, and (2) a more sophisticated extension to the Vickrey–Clarke–Groves (VCG) auction that preserves the truthful bidding property.

of clicks (actually observed in real time) and the per-click charge (determined during the auction).

Before we analyze a K-item auction, we will first study auctions with only $K = 1$ item. There are two main types of such one-item auctions: ascending-price and descending-price. These intuitive types of auction, in use since the Roman Empire, require a public venue for announcing the bids.

- In an **ascending price auction**, an auctioneer announces a base price, and then each bidder can raise her hand to bid a higher price. This price war keeps escalating until one bidder submits a price, no other bidder raises a hand, and the auctioneer calls out "gone." The last bidder is the winning bidder, and she pays the price she bid in the last round.
- In a **descending price auction**, an auctioneer announces a high price first, so high that no bidder is willing to accept it. The auctioneer then starts to lower the price, until there is one bidder who shouts out "OK." That bidder is allocated the item, and pays the price announced when she said "OK."

The alternative to a public venue is private disclosure of bids, called **sealed envelope** auctions. This is much more practical in many settings, including selling ad spaces by Google and auctioning goods on eBay. There are two types of such auctions, but it turns out that their results are essentially equivalent to the two types of open auctions we just discussed.

Each bid b_i is submitted by bidder i in a sealed envelope. All bids are then revealed simultaneously to the auctioneer, who will then decide

- the allocation, and
- how much to charge for each item.

The allocation part is easy: the highest bidder gets the item; but the amount charged can vary.

- In a **first price auction**, the winner pays the highest bid, i.e., her own bid.
- In a **second price auction**, the winner pays the second highest bid, i.e., the bid from the next highest bidder.

Second price auction sounds "wrong." If I know I will be paying the next highest bid, why not bid extremely high so that I can win the item, and then pay a much lower price for it? As it turns out, this intuition *itself* is wrong. The assumption of "much lower prices being bid by other bidders" does not hold when everyone engages in the same strategic thinking.

Instead, a second price auction is effectively equivalent to the highly intuitive ascending price auction, and can induce truthful-bidding behavior from the bidders. That is why second price auction is used so often, from auctioning major municipal projects to auctioning wireless spectrum.

Finally, we come back to auctions of K items (still with 1 seller and N bidders). If we follow the basic mechanism of second price auction, we obtain what is called the **Generalized Second Price** (GSP) for ad space auction: the ith ad space goes to the bidder that puts in the ith highest bid, and the charge, per clickthrough rate, is the $(i+1)$th bid. If the webpage layout shows the ads vertically, the advertiser in a given ad space is paying a price that is the same as the bid from the advertiser in the ad space *right below* hers. This simple method is used by Google in selling its ad spaces.

But it turns out GSP is *not* an auction that induces truthful bidding, and there can be many Nash equilibria if we analyze it as a game. An alternative is the **Vickrey–Clarke–Groves** (VCG) auction, which actually extends the second price auction's property of truthful bidding to multiple-item auctions. A VCG auction charges on the basis of negative externality, a principle that we will see many times throughout this book. The relationships between these types of auction are summarized in Figure 2.1.

Throughout the chapter, we focus on the simplest case, where there is a single round of bidding. In reality, there are multiple related bids going on at the same time, e.g., `www.banfflakelouise.com` may be bidding for multiple related keywords, such as "Banff," "Lake Louise," and "Canadian vacation," simultaneously. In a homework problem, we will go into a little more detail on one aspect of simultaneous auction in the context of spectrum auctioning.

2.2 A Long Answer

2.2.1 When do we need auctions?

No matter which format, an auction runs a resource-allocation process. It allocates items among bidders and sets the prices. We assume each bidder has a valuation of the item, and that the valuation is private and independent. **Private valuation** means that the value is unknown to others (all the other bidders and the auctioneer). **Independent valuation** means that one bidder's valuation does not depend on other bidders' valuations.

You will see that all assumptions are false, which is *why* we call them "assumptions" rather than "facts." But some assumptions are so false that the theory built on them loses predictive power. In this chapter, the private- and independent-valuation assumptions for auction will be used quite fruitfully, but it is still worthwhile to point out that they may be false in reality.

- In many instances, valuations often *depend* on each other, especially when there is a secondary market for you to resell the items. If you are bidding on a foreclosure house, you are probably dealing with dependent valuation.
- Valuations are sometimes *public*. In fact, some eBay bidding strategies attempt to reveal others' valuations, a particularly helpful strategy when you do not know how to value an item and you think other bidders might be experts who know more.
- Valuations in some cases are actually *not precisely known* even to the bidder herself.

There are several criteria used to compare the different outcomes of an auction.

- Seller's revenue, which clearly the seller would like to maximize. It often turns out that the revenue-maximizing strategy is very hard to characterize.
- The sum of payoffs received by all the bidders. This metric will become clearer as we model auctions as games. There is a tradeoff between seller revenue and bidders' payoff.
- Truthful bidding, a property of an auction where each bidder bids her true valuation.

2.2.2 Auctions as games

We can view an auction as a game.

1. The set of players is the set of N bidders, indexed by i.
2. The strategy is the bid b_i of each bidder, and each has a strategy space being the range of bids that she might put forward.
3. Each bidder's payoff function, U_i, is the difference between her valuation v_i of the item and the price p_i she has to pay, if she wins the auction:

$$U_i(\mathbf{b}) = v_i - p_i(\mathbf{b}).$$

On the other hand, the payoff is

$$U_i(\mathbf{b}) = 0$$

if bidder i loses the auction, since she is neither paying anything nor getting the item. Obviously, winning or losing the auction is also an outcome determined by $\mathbf{b}$.

Here, the coupling of the bidder's bidding choices is shown through the dependence of U_i on the entire *vector* $\mathbf{b}$, even though the valuation is independent: v_i depends only on i. Different auction rules lead to different $p_i(\mathbf{b})$; that is the mechanism-design part. By changing the rules of an auction, we can induce different bidding behaviors. For a given mechanism, each bidder will pick her b_i to maximize U_i:

$$b_i^* = \text{argmax}_{b_i} U_i(b_1, b_2, \ldots, b_N).$$

In the case of a first price (sealed envelope) auction, the price p_i which the bidder has to pay is simply her own bid value b_i, since that is the highest bid. So the payoff for bidder i, when winning, is $v_i - b_i$. From this bidder's perspective, she wants to pick the "right" b_i: not so big that the payoff is small (possibly zero) when winning, and not so small that she does not win the auction at all and receives 0 payoff. It is not easy to solve this optimization since it involves the strategies of all other bidders. But one thing is clear: she should bid less than her true valuation, for otherwise the best payoff she can receive is zero.

In the case of a second price (sealed envelope) auction, the price p_i one has to pay (when winning the auction) is the second highest price, i.e., the bid from say bidder j. The payoff is $v_i - b_j$ if bidder i wins, and 0 otherwise. In this game, it turns out that, *no matter* what other bidders might bid, each bidder can maximize her payoff by bidding her true valuation: $b_i = v_i$. Truthful bidding is a dominant strategy for the game. In other words, if the auction rule says that $p_i(\mathbf{b})$ is the second-highest entry in $\mathbf{b}$, then setting $b_i = v_i$ maximizes U_i for each bidder, no matter what the values of the rest of the entries in $\mathbf{b}$ are.

2.2.3 Single-item auction: Second price

There are several ways to view the somewhat counter-intuitive result of bidding behavior in a second price auction. Fundamentally, it is precisely the *decoupling* of the payoff amount (how much you gain if you win) from the auction result (whether you win at all) that induces each bidder to bid her true valuation. This is also what happens in the familiar format of an open auction with ascending price. Your bid determines how long you will stay in the price war. But when it stops, you pay the bid of the next-highest bidder plus a small amount capped by the minimum increment per new bid (unless you overbid much more than the minimum increment), for that is the auctioneer's announcement when the price war stops as a result of the next-highest bidder dropping out.

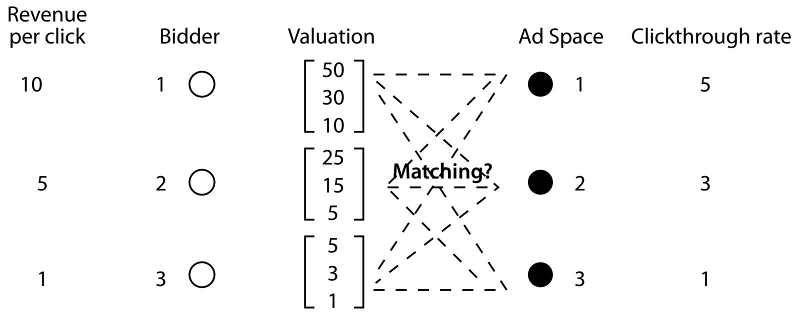

Figure 2.2 An example of three bidders and three ad spaces. Each bidder's valuation is a vector now. Each valuation vector is proportional to the vector of clickthrough rates $[C_1, C_2, C_3]$, where the proportionality constant is the average revenue per click, R, for that bidder of ad space. We assume that the clickthrough rates are independent of the content of the ad placed at each position. If the bidders and ad spaces are listed in descending order of their R and C, and each player bids true valuations, GSP simply matches "horizontally."

We can readily convince ourselves that truthful bidding is a dominant strategy. Say you want to bid your true valuation: $b = v$. Suppose that, as your advisor, I suggest lowering that bid to be something less than v. Call this new bid $\tilde{b}$. You should realize that such an action will change the outcome (auction result or payment amount) only if the next highest bid, say user 2's bid, b_2, is between b and $\tilde{b}$: $b > b_2 > \tilde{b}$. And in this case, you lose the auction, which you could have won and received a positive payoff of $v - p = v - b_2 = b - b_2 > 0$. So you would rather not take my advice to lower your bid.

Suppose instead I suggest raising your bid to be something more than v. Call this new bid $\hat{b}$. Again, such an action will change the outcome only if the highest bid, let us say user 2's bid, b_2, is in between b and $\hat{b}$: $\hat{b} > b_2 > b$. And in this case, you now win the auction, but receive a negative payoff, as you end up paying more than you value the item: $v - p = v - b_2 = b - b_2 < 0$. Had you bid b, you would have lost and received a payoff of 0, a better outcome than a negative payoff. So you would rather not take my advice to raise your bid.

Therefore, no matter what other bidders do, you would rather neither lower nor raise your bid. You would rather simply bid v.

This argument seals the math, but what is the intuition behind the math? For example, while it is clear why a first price auction is not truthful-bid-inducing, what about a third price auction? We will see in the Advanced Material that the intuition is that second price here can capture the negative externality, the "damage," caused by the winner to the other bidders.

2.2.4 Multiple-item auction: Generalized second price (GSP)

So far in this section, we have been examining auctions with only one item to sell. Search ad markets have multiple ad spaces to sell in each auction: multiple items facing multiple bidders. For simplicity, we assume that there is only one auction going on. Let us say there are three ad spaces in this auction, with an average clickthrough rates of 5, 3, and 1, as shown in Figure 2.2. And there are three advertisers (bidders for ad spaces), with different expected revenues per click, say 10, 5, and 1 (all in dollars). Then the valuation of each advertiser is as follows: the first advertiser has valuations of [50, 30, 10] for the three spaces, the second advertiser has valuations of [25, 15, 5], and the third advertiser has valuations of [5, 3, 1]. The job of a multiple-item auction is to assign one item to each advertiser.

If the number of advertisers and the number of ad spaces are different, some advertisers may get no ad space or some ad space will be left unsold. We will consider these cases as simple extensions in a homework problem.

We will later encounter other types of **bipartite graphs**, like the one we saw for the relation between the auctioned items and bidders in Figure 2.2. In a bipartite graph, all the nodes belong to either the left column or the right column, and all the links are only between the two columns but not within each column.

Now, back to multiple-ad-space auctions. A bidder, i.e., an advertiser, i sends in a bid b_i, indicating how much she is willing to pay per click. This is actually a significant simplification in an ad auction. Since there are multiple ad spaces, why not ask each advertiser to submit many bids, one bid per ad space, effectively presenting a scale of their preferences? We will encounter such vector preferences later, but for ad auction, the industry thought that a scalar representation of this vector preference suffices. Each entry in the valuation vector is just a multiple of this scalar.

So, advertiser i sends in a vector of bids $[b_iC_1, b_iC_2, \ldots, b_iC_K]$ for the K ad spaces with clickthrough rates $C_j, j = 1, 2, \ldots, K$. She may *not* pick b_i to be the same as the expected revenue per click.

In GSP, the ith ad space is allocated to the ith highest bidder. As we will see in the Advanced Material, this rule of ad space allocation is the same for VCG. But the amount each bidder is charged is different depending on whether GSP or VCG is used, and the strategic thinking in bidders' minds is consequently different too. In GSP, the charges are easy to determine: the ith ad space winner pays the per click price of the $(i + 1)$th highest bidder.

In summary, for GSP the following criteria apply.

- The "allocation part" is easy to see: advertiser i's bidding vector is proportional to the vector of clickthrough rates $[C_1, C_2, \ldots, C_K]$, where the proportionality constant is bid b_i. So the advertiser who sends in the ith highest bid for each click gets the ith most-valuable ad space. Advertiser i then drops out of future bidding after winning an ad space.
- The "charge part" is simply a straight-forward extension of the second price approach. (We could have also generalized the first price approach, as Overture did in the late 1990s, but soon realized that it was an unstable mechanism.)

Often, a minimum bid is also mandated, say, \$0.5. So in the above example in Figure 2.2, if all the advertisers bid their true valuations, advertiser 1 is matched to ad space 1, with a price of \$5 per click, advertiser 2 to ad space 2, with a price of \$1 per click, and advertiser 3 to ad space 3 with a price of \$0.5 per click, the minimum bid allowed.

We can also consider GSP as a game too. Despite the apparent similarities between GSP auctions (for multiple items) and second price auctions (for a single item), there are substantial differences. There can be multiple Nash equilibria in the GSP game, and it is possible that none of them involves truthful bidding. The example in the next section illustrates these properties of GSP. This is in sharp contrast to second price auction's desirable property: truthful bidding as a dominant strategy. For a multiple-item auction that preserves this property, we will have to wait until the Advanced Material, where we will describe the VCG auction. It is not "second price" that matters, but charging on the basis of the damage caused to other bidders.

2.3 Examples

2.3.1 Single-item auction on eBay

Let us start with an examle of single-item auctions. Founded in 1995 and with over 40 million users now, eBay runs online auctions for all kinds of goods. The eBay auction style largely follows the second price auction, but is not exactly the same. There are four main differences:

- A seller announces a start price, but can also choose to specify a secret *reserve price*: the minimum price below which she will not sell the good. Bidders are aware of the existence of the reserve price but not the actual value. This allows a seller to set a start price low enough to attract interest but still maintain a minimum revenue upon the actual sale of the item. In the rest of this section, for simplicity, we will assume the reserve price is the same as the start price. There is also a *minimal increment* of δ dollars: the winner pays the second-highest bid plus this δ (unless that exceeds her own bid, in which she just pays her own bid, the highest bid).
- An eBay auction is *not* sealed envelope. Some information about the current bids is continuously released to the public. This is particularly helpful when some bidders are not sure about the valuation of a good. It also generates more "fun" in the auction process. So what information is displayed? eBay looks at the price the winner would need to pay, had the auction been concluded right now, i.e., the smaller of (1) the current highest bid, b_1, and (2) the second highest bid, b_2, plus δ. This price is announced in public. The next bid has to exceed it by another minimum increment δ, which becomes the new *ask price* displayed. So, the announced price is $\min\{b_1, b_2 + \delta\} + \delta$.

- It has a fixed and publicly announced time horizon, e.g., 3 days. This *hard closing* rule changes bidding behavior. For example, many bidders choose to wait until the last 10 minutes to enter their bids, knowing that it is the time period that really matters. There are even third-party tools for automated "sniping," where bids are sent on your behalf in the last seconds before closing. It is advantageous to wait if you would like to surprise your competitors or to learn something about their valuations. It is advantageous not to wait if you would like to scare the competitors away with a very high bid to start with and avoid a dragged-out bidding war.
- It allows automated "proxy agent" bidding to simplify the user interface. A bidder can enter the *maximum bid* she is willing to put in, and then let the proxy run the course. When the bidder is no longer the highest bidder, yet the displayed ask price is less than or equal to this indicated maximum bid, a bid is automatically entered to take the ask price.

Here is an example of a timeline illustrating an eBay auction with three bidders: Alice, Bob, and Chris.

- *Start*
  Sam, the seller, lists a lamp for sale on eBay, with a start price of \$5.00 and a duration of 5 days. The minimum increment is \$1.00. The reserve price is set to be the same as the start price.
  Highest bid: n/a; Ask price: \$5.00.

- *Day 1*
  The first bid is from Alice, who uses a proxy agent with the maximum bid set to \$12.00. eBay therefore bids \$5.00 on Alice's behalf. The ask price becomes $\$5.00 + \$1.00 = \$6.00$.
  Highest bid: Alice, \$5.00; Ask price: \$6.00.

- *Day 2*
  The second bid is from Bob, who bids \$8.00 even though the ask price is \$6.00. eBay immediately raises bid to $\$8.00 + \$1.00 = \$9.00$ on Alice's behalf, since she is using proxy-agent bidding and the ask price is not greater than her maximum bid. The ask price becomes $\min\{\$9.00, \$8.00 + \$1.00\} + \$1.00 = \$10.00$.
  Highest bid: Alice, \$9.00; Ask price: \$10.00.

- *Day 3*
  Bob tries again, bidding \$10.50 this time. eBay immediately raises the bid to $\$10.50 + \$1.00 = \$11.50$ on Alice's behalf. The ask price becomes $\min\{\$11.50, \$10.50 + \$1.00\} + \$1.00 = \$12.50$.
  Highest bid: Alice, \$11.50; Ask price: \$12.50.

- *Day 4*
  Bob gets frustrated and raises his bid to \$17.50. The highest bidder now changes to Bob and the ask price becomes $\min\{\$17.50, \$11.50 + \$1.00\} + \$1.00 = \$13.50$.
  Highest bid: Bob, \$17.50; Ask price: \$13.50.

- *Day 5*
  It so happens that Chris enters the auction at the last moment and bids \$18.00, even though he does not know \$17.50 is the current highest bid. The ask price becomes $\min\{\$18.00, \$17.50 + \$1.00\} + \$1.00 = \$19.00$.
  Highest bid: Chris, \$18.00; Ask price: \$19.00.

- *End*
  Nobody takes the ask price before the auction terminates at the hard closing time announced before. The auction ends and Chris wins the lamp with a price of $\min\{\$18.00, \$17.50 + \$1.00\} = \$18.00$.

The bidding history is summarized in Table 2.1:

	Day 1	Day 2	Day 3	Day 4	Day 5
Alice	\$5.00	\$9.00	\$11.50	–	–
Bob	–	\$8.00	\$10.50	\$17.50	–
Chris	–	–	–	–	\$18.00
Ask price	\$6.00	\$10.00	\$12.50	\$13.50	\$19.00

Table 2.1 The bidding history and ask price evolution in an example of an eBay auction. There are three bidders, each using a different bidding strategy, to compete for a single item. The auction lasts 5 days.

2.3.2 Multiple-item GSP auction in Google

Now we move on to multiple-item auctions in the GSP style. Let us consider the bipartite graph in Figure 2.2 again.

- *Bidding*: Assume truthful bidding, i.e., $\mathbf{b}$ is the same $\mathbf{v}$ for each of the three bidders as shown in the graph. For example, bidder 1 bids \$50 (per hour) = \$10 per click $\times$ 5 clicks per hour, for ad space 1, \$30 for ad space 2, etc.
- *Auction outcome (matching, or allocation)*: By the GSP mechanism, the most valuable ad space is allocated to the highest bidder, which clearly is bidder 1 (since bidder 1 bids 50, bidder 2 bids 25, and bidder 3 bids 5 for this ad space). Similarly, ad space 2 is allocated to bidder 2, and ad space 3 to bidder 3. This matching is not surprising, since we have already ordered the bidders and the ad spaces in descending order.
- *Auction outcome (Charging, or pricing, or payment)*: Assume the actual number of clicks per hour is the same as the estimated clickthrough rate: bidder

1 pays the second-highest bid, which is \$5 (per click). So she pays $\$5 \times 5$ (the second 5 here refers to the clickthrough rate of 5 per hour for ad space 1)$=$ \$25 per hour for ad space 1. Bidder 2 payers \$1 per click. So she pays $\$1 \times 3 = \$3$ per hour for ad space 2. Bidder 3 pays just the minimum bid, e.g., \$0.5 per click as set by Google. So she pays $\$0.5 \times 1 = \$0.5$ per hour.

- *Revenue to the seller*: Google has collected a revenue of $\$25 + 3 + 0.5 = \$28.5$ per hour.
- *Payoff for each bidder*: Payoff is valuation v minus price p. So bidder 1's payoff is $\$10 - \$5 = \$5$ per click, or equivalently, $\$5 \times 5 = \$25$ per hour. Bidder 2's payoff is $\$5 - \$1 = \$4$ per click, or $\$4 \times 3 = \$12$ per hour. Bidder 3's payoff is $\$1 - 0.5 = \$0.5$ per click, or $\$0.5 \times 1 = \$0.5$ per hour.

  In summary, the total payoff is $\$25 + 12 + 0.5 = \$37.5$ per hour.

2.3.3 Another example of GSP

Suppose there are two ad spaces on a webpage and three bidders. An ad in the first space receives 400 clicks per hour, while the second space gets 200. Bidders 1, 2, and 3 have values per click of \$12, \$8, and \$4, respectively.

If all advertisers bid truthfully, then the bids are (in dollars) $[4800, 2400]$ from bidder 1, $[3200, 1600]$ from bidder 2, and $[1600, 800]$ from bidder 3. Bidder 1 wins the first ad space, paying \$8 per click, while bidder 2 wins the second space, paying \$4 per click. Bidder 3 does not get any ad space. If the actual clickthrough rates are the same as the estimated ones, the payments of bidders 1 and 2 are \$3200 and \$800, respectively. And the payoffs are \$1600 and \$800, respectively.

Truth-telling is indeed an equilibrium in this example, as you can verify that no bidder can benefit by changing her bids.

But, in general, truth-telling is not a dominant strategy under GSP. For example, consider a slight modification: the first ad space receives 400 clicks a day, and the second one 300. If all players bid truthfully, then bidder 1's payoff, as before, is $(\$12 - \$8) * 400 = \$1600$. If, instead, she shades her bid and bids only \$7 per click to get the second ad space, her payoff will be equal to $(\$12 - \$4) \times 300 = \$2400$, which is bigger than \$1600. Therefore, she would bid below her valuation and receive a higher payoff. The difference between the first and second ad spaces' clickthrough rates is simply not large enough relative to the difference in per-click payment for her to bid truthfully.

2.4 Advanced Material

2.4.1 VCG auction

As we just saw, the GSP auction does not guarantee truthful bidding for multiple-item auctions. If that property is desired, the proper generalization of second price auction is the VCG auction.

To search for the correct intuition, we revisit single-item second price auctions. Had the highest bidder not been there in the auction, the second-highest bidder would have obtained the item, while the other bidders face the same outcome of losing the auction. So the "damage" done by the highest bidder to the entire system (other than the highest bidder herself) is the valuation of the second highest bidder, which is exactly how much the highest bidder is charged. For example, suppose the three valuations are 10, 5, and 1, respectively from Alice, Bob, and Chris. If Alice were not there in the system, Bob would have won the item, and Chris would have lost the auction anyway. Now that Alice is in the system, Bob loses the chance to receive a valuation of 5, and Chris does not suffer any lost valuation. So the total valuation lost to the system (other than Alice) is 5, which is the price Alice pays according to the second price auction's rule.

The key idea to realize is that, in a second price auction, the winner is charged for how much her winning reduces the payoffs of the other bidders. This is another instance of internalizing negative externality. This time the negative externality is imposed by the winner on all the other bidders. It is this property that enables truthful bidding.

In Chapter 1, power control in wireless networks compensates for negative externality due to signal interference. We will later see negative externalities characterized and compensated for in voting systems, in tragedy of the commons, and in TCP congestion control. In multiple-item auctions, we compare two scenarios and take the difference as the quantification of the "damage" done by bidder i:

- The revenue generated by Google if bidder i were not in the system.
- The revenue generated by Google from all other bidders when bidder i is in the system.

In VCG auctions, there are three main steps, as summarized in Figure 2.3.

1. The first step is to invite each bidder to send in her bids, one for each of the items. In Google's case, they are proportional to a common scalar. Let v_{ij} be the value of the bid submitted to the seller by bidder i for item j. Now, these bids might not be the true valuations. But, as we will soon discover, by properly matching the bidders with the items and then charging the right prices, we can induce the bidders to submit the true valuations as $\{v_{ij}\}$. Again, auction design can be viewed as a mechanism design problem.
2. Second, an optimization is carried out by the seller. A **matching** is computed through this optimization, in which each item is matched to a bidder.

   For example, if we draw three horizontal lines from bidders to items in Figure 2.2, that would be a matching, denoted as $\{(1,1),(2,2),(3,3)\}$. This matching returns a total valuation of $50+15+1=66$ to the bidders.

   In VCG, we want the matching to maximize the total valuation of the *whole system*: we maximize $\sum_{(ij)} v_{ij}$ (where the sum is over all the matched

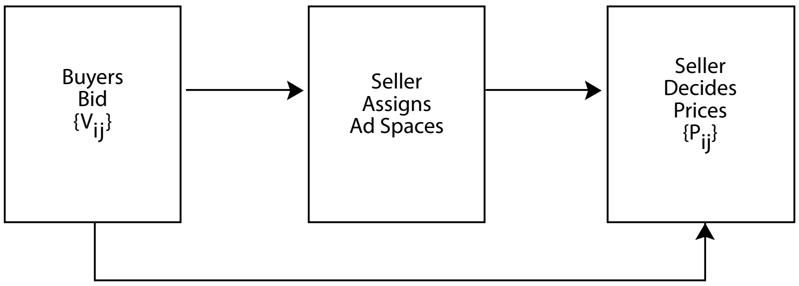

Figure 2.3 Three steps in VCG auctions. First, each bidder sends in a vector of bids for the K items. Then the seller solves an optimization problem of finding the matching between bidders and items (and computes the corresponding maximized value V for the system). Finally, given the matching, the seller determines the price p_{ij} that bidder j pays for item i that she is matched to, on the basis of the amount of negative externality caused by this matching.

pairs) over all the possible matchings. For example, a different matching of $\{(1,2),(2,3),(3,1)\}$ would return a total valuation of $\sum_{(ij)} v_{ij} = 30+5+5 = 40$, an inferior matching to $\{(1,1),(2,2),(3,3)\}$, as illustrated in Figure 2.4.

Denote by V the resulting maximized total value. Suppose item j is allocated to bidder i in this matching. Obviously, we can write V as the sum of the value v_{ij} and $\hat{V}_{i\leftarrow j}$, the sum of the values of all the *other* (item, bidder) pairs in the matching given that item j goes to bidder i:

$$V = v_{ij} + \hat{V}_{i\leftarrow j}.$$

This notation will become useful soon.

3. Matching decides the allocation, and pricing is determined only after the matching. For a given matching, each pair of (item, bidder) is charged a price of p_{ij}, which is the amount of damage caused by this matching. So, how do we quantify the damage done by a bidder i getting item j?

   Imagine two alternative systems. (a) A system *without* bidder i, and there is a corresponding $V_{no\ i}$ as the maximum total valuation for everyone else. (b) A system *with* bidder i, who will get item j, and the corresponding $\hat{V}_{i\leftarrow j}$ as the total maximum valuation for everyone else. The difference between these two valuations, $V_{no\ i} - V_{i\leftarrow j}$, is the "damage" caused to everyone else, and that is the price p_{ij} the seller charges user i for being matched to item j:

$$p_{ij} = V_{no\ i} - \hat{V}_{i\leftarrow j}. \tag{2.1}$$

   This completes the description of a VCG auction.

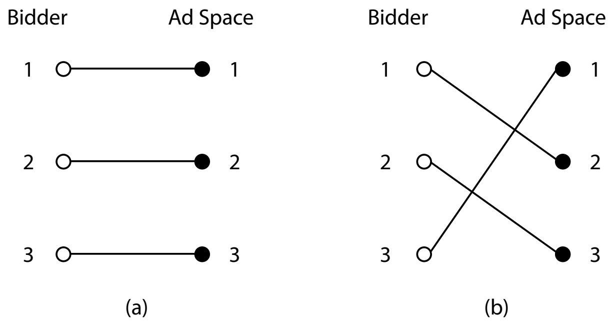

Figure 2.4 Two of the many possible matchings between bidders (on the left) and ad spaces (on the right) of a multi-item auction's bipartite graph. The weight of link (ij) is $v_{ij} = R_iC_j$. A maximum weight matching selects a subset of those links so that each bidder is matched to an item, and the sum of these links' weights is the largest possible among all matchings. With the parameter values in Figure 2.2, matching (a) turns out to maximize $\sum_{(ij)} v_{ij}$ (and is thus chosen in step 2 of a VCG auction) while matching (b) does not.

2.4.2 An example

As in the GSP example we just saw in the last section, suppose there are two ad spaces on a webpage and three bidders. An ad in the first space receives 400 clicks per hour, while the second space gets 200. Bidders 1, 2, and 3 have valuations per click of \$12, \$8, and \$4, respectively. Suppose the bids are the true valuations for now; we will later *prove* that it is indeed a dominant strategy in the auction game.

The matching by VCG happens to be the same as by GSP in this case: bidder 1 gets the first ad space, and bidder 2 the second, with bidder 3 not matched to any ad space. The second bidder's payment is still \$800 because the damage it causes to the system is that bidder 3 lost the second ad space, thus $\$4 \times 200 = \$800$ value. However, the payment of bidder 1 is now \$2400: \$800 for the damage to bidder 3 and \$1600 for the damage to bidder 2: bidder 2 moves from position 1 to position 2, thus causing her $(400-200) = 200$ clicks per day at the valuation of \$8 a click.

In this example, the seller's revenue under VCG is $\$2400+\$800 = \$3200$, lower than that under GSP ($\$3200 + \$800 = \$4000$), if the advertisers bid truthfully in both cases.

2.4.3 Truthful bidding

So back to this question: what would each (rational) bidder i bid for item j in a VCG auction? We claim it must be the true valuation, i.e., the expected revenue per click times the estimated clickthrough rate.

Suppose that was not true, and bidder i bids some other number. Again, it is important to realize that this would make a difference in a VCG auction's matching only if it resulted in bidder i getting a different item, say item h, e.g., as in Figure 2.4(b). Obviously,

$$v_{ij} + \hat{V}_{i \leftarrow j} \geq v_{ih} + \hat{V}_{i \leftarrow h},$$

since the left side is V, the *maximized* total valuation over *all* possible matchings. Subtracting $V_{no\ i}$ from both sides,

$$v_{ij} + \hat{V}_{i \leftarrow j} - V_{no\ i} \geq v_{ih} + \hat{V}_{i \leftarrow h} - V_{no\ i},$$

and using the definition of p_{ij} in (2.1), we have

$$v_{ij} - p_{ij} \geq v_{ih} - p_{ih},$$

i.e., the payoff of bidding true valuation is higher. This argument shows that VCG induces truthful bidding as a dominant strategy for each bidder.

2.4.4 Other considerations

What about the seller's perspective: to maximize (over all the possible matchings) $\sum_{(ij)} p_{ij} C(j)$, the sum of per-click price times the estimated clickthrough rate across the (item, bidder) pairs matched. As we just saw in the example, GSP may generate more revenue than VCG, or less, depending on the problem parameters' values. If the actual clickthrough rate, as a function of the ad space location j, also depends on who is the winning bidder i, i.e., $C(j)$ becomes $C_i(j)$ and depends on i, then the seller's revenue maximization problem becomes even more challenging.

Which one to use: GSP or VCG? Google uses GSP, while many web 2.0 companies use VCG. Companies like AppNexus run variants of VCG auctions on large-scale (billions of auctions each day), real-time (millisecond matching), and user-profile-based targeted-ad placement. The comparison between GSP and VCG needs to include various considerations:

- GSP is simpler for Google to explain to advertisers than VCG.
- While GSP is simple when $C(j)$ is independent of winner i, it becomes more complicated when the clickthrough rate of an ad space depends on the actual ad placed there.
- While VCG guarantees truthful bidding, that might not be the case if there are multiple auctions held in parallel across many websites.
- If Google switches from GSP to VCG, the advertisers may act irrationally. They may ignore this change, continue to shade their bids, and use the same bids as before. For the same set of bids, VCG revenue is lower than GSP revenue. Relative to the potential benefit, the cost of transitioning from GSP to VCG may be too high for Google.

What is clear from this chapter is that different transaction mechanisms can induce very different behaviors from people. People react to different prices with different demands, react to different allocation methods with different strategies, and react to different expectations of tomorrow's events with different actions today. More generally, our design of a network and its functions may induce different optimization problems for the operators or the individual players. This is a recurring theme throughout this book.

Summary

Box 2 Second price auctions

Auctions allocate items among competing bidders. Different mechanisms of allocation and charging leads to different bidding behaviors and strategic equilibira. Charging based on externality, like in a second price auction of a single item or in a VCG auction of multiple items, induces truthful bidding as a dominant strategy in bidders' strategic thinking.

Further Reading

Auction theory is a well-studied discipline in economics and in computer science. There are many variations of the basic auctions we covered here: reverse auction with one bidder and many sellers, double auctions with many bidders and many sellers, and multiple-winner auctions. Researchers have also studied topics like revenue maximization strategies and collusion among bidders.

1. The following is a standard textbook on the subject:
V. Krishna, *Auction Theory*, 2nd edn., Academic Press, 2009.

2. Another insightful survey with special emphasis on applications is
P. Milgrom, *Putting Auction Theory to Work*, Cambridge University Press, 2004.

3. The following book provides an in-depth discussion of eBay auctions and the surprising behaviors found there:
K. Steiglitz, *Snipers, Shills, and Sharks*, Princeton University Press, 2007.

4. The following short paper provides a concise survey of the key ideas in Google advertising and sponsored search:
H. R. Varian, "The economics of Internet search," *Rivista di Politica Economica*, vol. 96, no. 6, pp. 9–23, 2006.

5. A more technical discussion focusing on GSP can be found here, especially the viewpoint of auctions as a dynamic game:

B. Edelman, M. Ostrovsky, and M. Schwarz, "Internet advertising and the generalized second-price auction: Selling billions of dollars worth of keywords," *The American Economic Review*, vol. 97, no. 1, pp. 242–259, 2007.

Problems

2.1 *A simple ad space auction* ⋆

Three advertisers (1, 2, 3) bid for two ad spaces (A, B). The average revenues per click are \$6, \$4, \$3 for the bidders, respectively, and the clickthrough rate of the ad spaces are 500, 300 clicks per hour respectively.

(a) Draw the bipartite graph with nodes indicating advertisers/ad spaces and edges indicating values per hour. Indicate the maximum matching with bold lines.

(b) Assume a GSP auction with truthful bidding, what is the result of the auction in terms of the allocation, the prices charged, and the payoffs received?

2.2 *eBay Auction* ⋆⋆

Alice lists a lamp for sale on eBay via auction with both the start price and reserve price set to \$7.00 and a duration of 5 days. The minimal increment is \$0.25 and the following events happen during the auction:

- *Day 1* Bidder 1 uses a proxy agent, setting the maximum bid up to \$11.00.
- *Day 2* Bidder 2 bids \$9.25.
- *Day 3* Bidder 3 uses a proxy agent, setting the maximum bid up to \$17.25.
- *Day 4* Bidder 2 bids \$13.65.
- *Day 5* Bidder 1 bids \$27.45.

List the bidding history of all three bidders over each day of the auction. Who is the winner and what price does she pay?

2.3 *More items than bidders* ⋆

Alice and Bob are bidding for three ad slots on a webpage, and one bidder can win at most one slot. Suppose the clickthrough rates are 500, 300, and 200 per hour, respectively. Assume that Alice receives \$$r$ per click.

(a) Denote by b_1 and b_2 the bids by Alice and Bob respectively. In GSP auction, discuss Alice's payoff in terms of b_1 and b_2.

(b) Does Alice have a dominant strategy? If so, what is it?

2.4 *Reverse auction* ⋆

Reverse auction is a type of auction where there are multiple sellers and only one bidder. The roles of bidders and sellers are reversed, that is, sellers lower their bids during auction and the one with the lowest bid sells her item.

Suppose there are three sellers in a reverse auction with one bidder. Denote b_i as the price seller i bids, and v_i as the value seller i attaches to the item.

(a) In the case of second price auction, what is the payoff function for seller i, as a function of b_1, b_2, and b_3?

(b) Is truthful bidding a dominant strategy?

2.5 *Spectrum auction and package bidding* ⋆⋆

Wireless cellular technologies rely on spectrum assets. Around the world, auctions have emerged as the primary means of assigning spectrum licenses to companies wishing to provide wireless communication services. For example, from July 1994 to July 2011, the US Federal Communications Commission (FCC) conducted 92 spectrum auctions, raising over $60 billion for the US Treasury, and assigned thousands of licenses to hundreds of firms to different parts of the spectrum and different geographic regions of the country.

The US FCC uses **simultaneous ascending auction**, in which groups of related licenses are auctioned simultaneously and the winner pays the highest bid. The British OfCom, in contrast, runs **package bidding**, where each potential spectrum bidder can bid on a joint set of frequency bands.

Among the many issues involved in spectrum auctioning is the debate between simultaneous ascending auction and package bidding auction. We will illustrate the inefficiency resulting from disallowing package bidding in a toy example. The root cause for this inefficiency is "bidder-specific complementarity" and the lack of competition.

Suppose that there are two bidders for two adjacent seats in a movie theater. Bidder 1 is planning to watch the movie together with her spouse as part of a date. She values the two spots *jointly* at $15, and a single spot is worth nothing. Bidder 2 plans to watch the movie by himself, and values each seat at $10, and the two seats together at $12 (since it is a little nicer to have no one sitting next to him on one side of his seat).

(a) Assume a simultaneous ascending auction is used for the seats, and Bidder 1 correctly guesses that Bidder 2 values $10 for one seat and $12 for two seats together. What strategy will Bidder 1 take? What is the result of the auction, in terms of the allocation, the price charged, and the payoffs received?

(b) Repeat part (a) but now assume package bidding is used. In particular, Bidder 1 can bid on a package consisting of both seats. Explain the differences with (a).

3 How does Google rank webpages?

3.1 A Short Answer

Now we turn to the other links you see on a search-result webpage; not the ads or sponsored search results, but the actual ranking of webpages by search engines such as Google. We will see that, each time you search on `www.google.com`, Google solves a very big system of linear equation to rank the webpages.

The idea of embedding links in text dates back to the middle of the last century. As the Internet scaled up, and with the introduction of the web in 1989, the browser in 1990, and the web portal in 1994, this vision was realized on an unprecedented scale. The network of webpages is huge: somewhere between 40 billion and 60 billion according to various estimates. And most of them are connected to each other in a giant component of this network. It is also sparse: most webpages have only a few hyperlinks pointing inward from other webpages or pointing outward to other webpages. Google search organizes this huge and sparse network by ranking the webpages.

More important webpages should be ranked higher. But how do you quantify *how* important a webpage is? Well, if there are many other important webpages pointing towards webpage A, A is probably important. This argument implicitly assumes two ideas:

- Webpages form a network, where a webpage is a node, and a hyperlink is a *directed* link in the network: webpage A may point to webpage B without B pointing back to A.
- We can turn the seemingly circular logic of "important webpages pointing to you means you are important" into a set of equations that characterize the *equilibrium* (a fixed-point equilibrium, not a game-theoretic Nash equilibrium) in terms of a *recursive definition* of "importance." This importance score will then act as an approximation of the ultimate test of search engines: how useful a user finds the search results.

As mentioned in Chapter 1, a network consists of both a *topology* and *functionalities*. Topology is often represented by a graph and various matrices, several of which will be introduced in this chapter and a few more in later chapters. And we will assume some models of the "search and navigation" functionality in this chapter.

Suppose there are N webpages. Each webpage i has O_i **outgoing links** and I_i **incoming links**. We cannot just count the number of webpages pointing to a given webpage A, because that number, the **in-degree** of the node in the hyperlinked graph, is often not the right measure of importance.

Let us denote the "importance score" of each webpage by π_i. If important webpages point to webpage A, maybe webpage A should be important too, i.e., $\pi_A = \sum_{i \to A} \pi_i$, where the sum is taken over all the webpages pointing to A. However, this is not quite right either, since node i may be pointing to many other nodes in this graph, and that means each of these nodes receives only a small portion of node i's importance score.

Let us assume that each node's importance score is *evenly* distributed across all the outgoing links from that node, i.e., each of the outgoing neighbors of node i receives π_i/O_i importance score. Now each node's importance score can also be written as the sum of the importance scores received *from* all of the incoming neighbors, indexed by j, e.g., for node 1,

$$\sum_{j \to 1} \frac{\pi_j}{O_j}.$$

If this sum is indeed also π_1, we have *consistency* of the scores. But it is not clear whether we can readily compute these scores, or even whether there is a consistent set of scores at all.

It turns out that, with a couple of modifications to the basic idea above, there is always a unique set of consistent scores, denoted as $\{\pi_i^*\}$. These scores determine the ranking of the webpages: the higher the score, the higher the webpage is ranked.

For example, consider a very small graph with just four webpages and six hyperlinks, shown in Figure 3.1. This is a directed graph where each node is a webpage and each link a hyperlink. A consistent set of importance scores turns out to be (0.125, 0.125, 0.375, 0.375): webpages 3 and 4 are more important than webpages 1 and 2. In this small example, it so happens that webpages 3 and 4, linking each other, push both webpages' rankings higher.

Intuitively, the scores make sense. First, by symmetry of the graph, webpages 1 and 2 should have the same importance score. We can view webpages 3 and 4 as if they form one webpage first, a supernode 3+4. Since node 3+4 has two incoming links, and each of nodes 1 and 2 has only one incoming link, node 3+4 should have a higher importance score. Since node 3 points to node 4 and vice versa, these two nodes' importance scores mix into an equal division at equilibrium. This line of reasoning qualitatively explains the actual scores we see.

But how do we calculate the exact scores? In this small example, it boils down to two simple linear equations. Let the score of node 1 (and 2) be x, and that of node 3 (and 4) be y. Looking at node 1's incoming links, we see that there is only one such link, coming from node 4 that points to three nodes. So we know

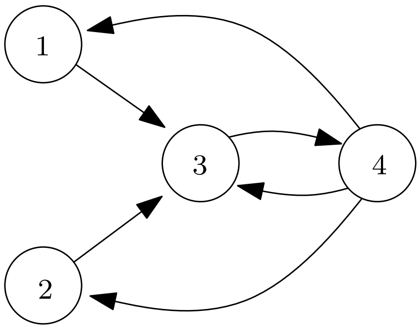

Figure 3.1 A simple example of importance score with four webpages and six hyperlinks. It is a small graph with much symmetry, leading to a simple calculation of the importance scores of the nodes.

$x = y/3$. By normalization, all scores must add up to $2x + 2y = 1$. So we have $x = 0.125$ and $y = 0.375$.

Now, how do we compute this set of consistent scores in a large, sparse, general graph of hyperlink connectivity?

3.2 A Long Answer

In any search engine, there are two main activities constantly occurring behind the scenes: (1) crawling the hyperlinked web space to get the webpage information, and (2) indexing this information into concise representations and storing the indices.

When you search in Google, it triggers a ranking procedure that takes into account two main factors:

- a **relevance score**: how relevant to the search the content is on each webpage, and
- an **importance score**: how important the webpage is.

It is the composite score of these two factors that determines the ranking. We focus on the importance score, since that usually determines the order of the top few webpages in any reasonably popular search, and has a tremendous impact on how people obtain information and how online businesses generate traffic.

We will be constructing several related matrices: $\mathbf{H}, \hat{\mathbf{H}}$, and $\mathbf{G}$, step by step (this matrix $\mathbf{G}$ is not the channel gain matrix of Chapter 1; it denotes the Google matrix in this chapter). Eventually, we will be computing an eigenvector of $\mathbf{G}$ as the importance-score vector. Each matrix is $N \times N$, where N is the number of the relevant webpages. These are extremely large matrices, and we will discuss the computational challenge of scaling-up in the Advanced Material.

3.2.1 Constructing $\mathbf{H}$

The first matrix we define is $\mathbf{H}$: its (i, j)th entry is $1/O_i$ if there is a hyperlink from webpage i to webpage j, and 0 otherwise. This matrix describes the network topology: which webpages point to which. It also evenly spreads the importance of each webpage among its outgoing neighbors, or the webpages that it points to.

Let π be an $N \times 1$ column vector denoting the importance scores of the N webpages. We start by guessing that the consistent score vector is $\mathbf{1}$, simply a vector of 1s because each webpage is equally important. So we have an initial vector $\pi[0] = \mathbf{1}$, where 0 denotes the 0th iteration, i.e., the initial condition.

Then, we multiply π^T on the right by the matrix $\mathbf{H}$. (By convention, a vector is a column vector. So when we multiply a vector on the *right* by a matrix, we put the transpose symbol T on top of the vector.) You can write out this matrix multiplication, and see that this is spreading the importance score (from the last iteration) evenly among the outgoing links, and re-calculating the importance score of each webpage in this iteration by summing up the importance scores from the incoming links. For example, $\pi_1[2]$ (for webpage 1 in the second iteration) can be expressed as the following weighted sum of importance scores from the first iteration:

$$\pi_1[2] = \sum_{j \to 1} \frac{\pi_j[1]}{O_j},$$

i.e., the inner-product of π vector from the previous iteration and the first column of $\mathbf{H}$:

$$\pi_1[2] = (\pi[1])^T(\text{column 1 of } \mathbf{H}).$$

Similarly,

$$\pi_i[2] = (\pi[1])^T(\text{column } i \text{ of } \mathbf{H}), \quad \forall i.$$

If we index the iterations by k, the update at each iteration is simply

$$\pi^T[k] = \pi^T[k-1]\mathbf{H}. \tag{3.1}$$

Again, we followed the (visually rather clumsy) convention in this research field that defined $\mathbf{H}$ such that the update is a multiplication of row vector π^T by $\mathbf{H}$ from the right.

Since the absolute values of the entries in π do not matter, only the ranked order, we can also normalize the resulting π vector so that its entries add up to 1.

Now the quesetion is: Do the iterations in (3.1) converge? Is there a K sufficiently large that, for all $k \geq K$, the $\pi[k]$ vector is arbitrarily close to $\pi[k-1]$ (no matter what the initial guess $\pi[0]$ is)? If so, we have a way to compute a consistent score vector as accurately as we want.

But the answer is "not quite yet." We need two adjustments to $\mathbf{H}$.

3.2.2 Constructing $\hat{\mathbf{H}}$

First, some webpages do not point to any other webpages. These are "dangling nodes" in the hyperlink graph. For example, in Figure 3.2, node 4 is a dangling

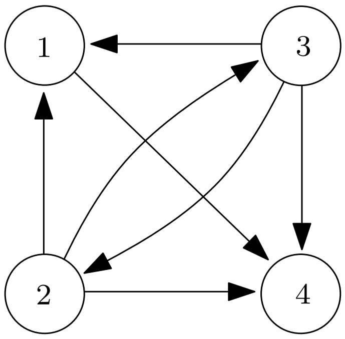

Figure 3.2 A network of hyperlinked webpages with a dangling node 4.

node, and its row is all 0s in the **H** matrix:

$$\mathbf{H} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1/3 & 0 & 1/3 & 1/3 \\ 1/3 & 1/3 & 0 & 1/3 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

There are no consistent scores. To see this, we write out the system of linear equations $\pi^T = \pi^T \mathbf{H}$:

$$\begin{cases} \frac{1}{3}(\pi_2 + \pi_3) = \pi_1 \\ \frac{1}{3}\pi_3 = \pi_2 \\ \frac{1}{3}\pi_2 = \pi_3 \\ \pi_1 + \frac{1}{3}(\pi_2 + \pi_3) = \pi_4. \end{cases}$$

Solving these equations gives $\pi_1 = \pi_2 = \pi_3 = \pi_4 = 0$, which violates the normalization requirement $\sum_i \pi_i = 1$.

One solution is to replace each row of 0s, like the last row in **H** above, with a row of $1/N$. Intuitively, this is saying that even if a webpage does not point to any other webpage, we will force it to spread its importance score evenly among all the webpages out there.

Mathematically, this amounts to adding the matrix $\frac{1}{N}(\mathbf{w}\mathbf{1}^T)$ to **H**, where **1** is simply a vector of 1s, and **w** is an indicator vector with the ith entry being 1 if webpage i points to no other webpages (a dangling node) and 0 otherwise (not a dangling node). This is an *outer product* between two N-dimensional vectors, which leads to an $N \times N$ matrix. For example, if $N = 2$ and $\mathbf{w} = [1\ 0]^T$, we have

$$\frac{1}{2}\begin{pmatrix} 1 \\ 0 \end{pmatrix}(1\ 1) = \begin{pmatrix} 1/2 & 1/2 \\ 0 & 0 \end{pmatrix}.$$

This new matrix we add to **H** is clearly simple. Even though it is big, $N \times N$, it is actually the same vector **w** repeated N times. We call it a **rank-1 matrix**.

The resulting matrix,

$$\hat{\mathbf{H}} = \mathbf{H} + \frac{1}{N}(\mathbf{w}\mathbf{1}^T),$$

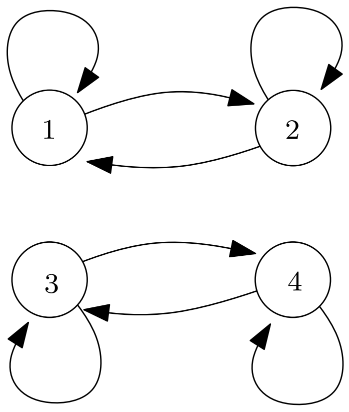

Figure 3.3 A network of hyperlinked webpages with multiple consistent score vectors.

has non-negative entries and each row adds up to 1. So we can think of each row as a probability vector, with the (i, j)th entry of $\hat{\mathbf{H}}$ indicating the probability that, if you are currently on webpage i, you will click on a link and go to webpage j.

Well, the structure of the matrix says that you are equally likely to click on any link shown on a webpage, and, if there is no link at all, you will be equally likely to visit any other webpage. Such behavior is called a **random walk on graphs** and can be studied as **Markov chains** in probability theory. Clearly this does not model web browsing behavior exactly, but it does strike a quite effective balance between the *simplicity* of the model and the *usefulness* of the resulting webpage ranking. We will see a similar model for influence in social networks in Chapter 8.

3.2.3 Constructing $\mathbf{G}$

We mentioned that there were two issues with $\mathbf{H}$. The second is that there might be *many* consistent score vectors, all compatible with a given $\hat{\mathbf{H}}$. For example, for the graph in Figure 3.3, we have

$$\mathbf{H} = \begin{bmatrix} 1/2 & 1/2 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 1/2 & 1/2 \end{bmatrix}.$$

Different choices of $\pi[0]$ result in different π^*, which are all consistent. For example, if $\pi[0] = [1\ 0\ 0\ 0]^T$, then $\pi^* = [0.5\ 0.5\ 0\ 0]^T$. If $\pi[0] = [0\ 0.3\ 0.7\ 0]^T$, then $\pi^* = [0.15\ 0.15\ 0.35\ 0.35]^T$.

One solution to this problem is to add a little *randomization* to the iterative procedure and the recursive definition of importance. Intuitively, we say there is a chance of $(1 - \theta)$ that you will jump to some other random webpage, without clicking on any of the links on the current webpage.

Mathematically, we add yet another matrix $\frac{1}{N}\mathbf{1}\mathbf{1}^T$, a matrix of 1s scaled by $1/N$ (clearly a simple, rank-1 matrix), to $\hat{\mathbf{H}}$. But this time it is a *weighted* sum, with a weight $\theta \in [0, 1]$. $(1-\theta)$ quantifies how likely it is that you will randomly jump to some other webpage. The resulting matrix is called the **Google matrix**:

$$\mathbf{G} = \theta\hat{\mathbf{H}} + (1-\theta)\frac{1}{N}\mathbf{1}\mathbf{1}^T. \tag{3.2}$$

Now we can show that, independently of the initialization vector $\pi[0]$, the iterative procedure:

$$\pi^T[k] = \pi^T[k-1]\mathbf{G} \tag{3.3}$$

will converge as $k \to \infty$, and converge to the unique vector π^* representing the consistent set of importance scores. Obviously, π^* is the left eigenvector of $\mathbf{G}$ corresponding to the eigenvalue of 1:

$$\pi^{*T} = \pi^{*T}\mathbf{G}. \tag{3.4}$$

One can then normalize π^*: take $\pi_i^*/\sum_j \pi_j^*$ as the new value of π_i^*, and rank the entries in descending order, before outputting them on the search result webpage in that order. The matrix $\mathbf{G}$ is designed such that there is a unique solution to (3.4) and that (3.3) converges from any initialization.

However you compute π^*, taking (the normalized and ordered version of) π^* as the basis of ranking is called the **PageRank algorithm**. Compared with DPC for wireless networks in Chapter 1, the matrix $\mathbf{G}$ in PageRank is much larger, but we can afford a centralized computation.

3.3 Examples

Consider the network in Figure 3.4 with eight nodes and sixteen directional links. We have

$$\mathbf{H} = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & 1/2 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 & 0 & 0 & 0 & 1/2 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 0 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 0 & 1/2 & 0 & 0 \\ 1/3 & 0 & 0 & 1/3 & 0 & 0 & 1/3 & 0 \end{bmatrix}.$$

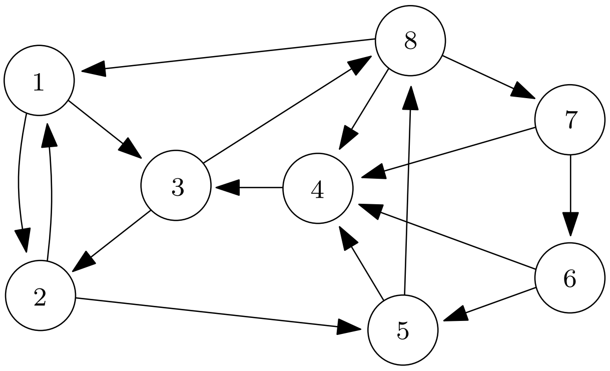

Figure 3.4 An example of the PageRank algorithm with eight webpages and sixteen hyperlinks. Webpage 3 is ranked the highest even though webpage 4 has the largest in-degree. Importance scores computed by PageRank can be quite different from node degrees.

Here $\hat{\mathbf{H}} = \mathbf{H}$ since there is no dangling node. Taking $\theta = 0.85$, we have

$$\mathbf{G} = \begin{bmatrix} 0.0188 & 0.4437 & 0.4437 & 0.0188 & 0.0188 & 0.0188 & 0.0188 & 0.0188 \\ 0.4437 & 0.0188 & 0.0188 & 0.0188 & 0.4437 & 0.0188 & 0.0188 & 0.0188 \\ 0.0188 & 0.4437 & 0.0188 & 0.0188 & 0.0188 & 0.0188 & 0.0188 & 0.4437 \\ 0.0188 & 0.0188 & 0.8688 & 0.0188 & 0.0188 & 0.0188 & 0.0188 & 0.0188 \\ 0.0188 & 0.0188 & 0.0188 & 0.4437 & 0.0188 & 0.0188 & 0.0188 & 0.4437 \\ 0.0188 & 0.0188 & 0.0188 & 0.4437 & 0.4437 & 0.0188 & 0.0188 & 0.0188 \\ 0.0188 & 0.0188 & 0.0188 & 0.4437 & 0.0188 & 0.4437 & 0.0188 & 0.0188 \\ 0.3021 & 0.0188 & 0.0188 & 0.3021 & 0.0188 & 0.0188 & 0.3021 & 0.0188 \end{bmatrix}.$$

Initializing $\pi[0] = [1/8\ 1/8\ \ldots 1/8]^T$, iteration (3.3) gives

$$\begin{aligned} \pi[1] &= [0.1073\ 0.1250\ 0.1781\ 0.2135\ 0.1250\ 0.0719\ 0.0542\ 0.1250]^T \\ \pi[2] &= [0.1073\ 0.1401\ 0.2459\ 0.1609\ 0.1024\ 0.0418\ 0.0542\ 0.1476]^T \\ \pi[3] &= [0.1201\ 0.1688\ 0.2011\ 0.1449\ 0.0960\ 0.0418\ 0.0606\ 0.1668]^T \\ \pi[4] &= [0.1378\ 0.1552\ 0.1929\ 0.1503\ 0.1083\ 0.0445\ 0.0660\ 0.1450]^T \\ \pi[5] &= [0.1258\ 0.1593\ 0.2051\ 0.1528\ 0.1036\ 0.0468\ 0.0598\ 0.1468]^T \\ \pi[6] &= [0.1280\ 0.1594\ 0.2021\ 0.1497\ 0.1063\ 0.0442\ 0.0603\ 0.1499]^T \\ &\vdots \end{aligned}$$

and $\pi^* = [0.1286\ 0.1590\ 0.2015\ 0.1507\ 0.1053\ 0.0447\ 0.0610\ 0.1492]^T$, to four decimal places. Therefore, the ranked order of the webpages are: 3, 2, 4, 8, 1, 5, 7, 6.

The node with the largest in-degree, i.e., with the largest number of links pointing to a node, is node 4, which is *not* ranked the highest. This is in part because its importance score is spread exclusively to node 3. As we will see again in Chapter 8, there are many more useful metrics measuring node importance than just the degree.

3.4 Advanced Material

3.4.1 Generalized PageRank and some basic properties

The Google matrix $\mathbf{G}$ can be generalized if the randomization ingredient is more refined. First, instead of the matrix $\frac{1}{N}\mathbf{1}\mathbf{1}^T$, we can add the matrix $\mathbf{1}\mathbf{v}^T$ (again, the outer product of two vectors), where $\mathbf{v}$ can be *any* probability distribution. Certainly, $\frac{1}{N}\mathbf{1}^T$ is a special case of that.

We can also generalize the dangling-node treatment: instead of adding $\frac{1}{N}\mathbf{w}\mathbf{1}^T$ to $\mathbf{H}$, where $\mathbf{w}$ is the indicator vector of dangling nodes, we can add $\mathbf{w}\mathbf{v}^T$. Again, $\frac{1}{N}\mathbf{1}$ is a special case of $\mathbf{v}$.

Now, the Google update equation can be written in the long form (not using the shorthand notation $\mathbf{G}$) as a function of the given webpage connectivity matrix $\mathbf{H}$, the vector $\mathbf{w}$ indicating the dangling webpages, and the two algorithmic parameters; scalar θ and vector $\mathbf{v}$:

$$\pi^T\mathbf{G} = \theta\pi^T\mathbf{H} + \pi^T(\theta\mathbf{w} + (1-\theta)\mathbf{1})\mathbf{v}^T. \tag{3.5}$$

You should verify that the above equation is indeed the same as (3.3).

There are many viewpoints to further interpret (3.3) and connect it to matrix theory, to Markov chain theory, and to linear systems theory. For example,

- π^* is the left eigenvector corresponding to the dominant eigenvalue of a positive matrix;
- it represents the so-called stationary distribution of a Markov chain whose transition probabilities are in $\mathbf{G}$; and
- it represents the equilibrium of an economic growth model according to $\mathbf{G}$ (more on this viewpoint is given later in this section).

The major operational challenges of running the seemingly simple update (3.3) are *scale* and *speed*: there are billions of webpages and Google needs to return the results almost instantly.

Still, the power method (3.3) offers many numerical advantages compared with a direct computation of the dominant eigenvector of $\mathbf{G}$. First, (3.3) can be carried out by multiplying a vector by the sum of $\mathbf{H}$ and two rank-1 matrices. This is numerically simple: $\mathbf{H}$ is very large but also very *sparse*: each webpage usually links to just a few other webpages, so almost all the entries in $\mathbf{H}$ are zero. Multiplying by rank-1 matrices is also easy. Furthermore, at each iteration, we only need to store the current π vector.

While we have not quantified the speed of convergence, it is clearly important to speed up the computation of π^*. As we will see again in Chapter 8, the convergence speed in this case is governed by the second-largest eigenvalue

$\lambda_2(\mathbf{G})$ of $\mathbf{G}$, which can be shown to be approximately θ here. So this parameter θ controls the tradeoff between convergence speed and the relevance of the hyperlink graph in computing the importance scores: smaller θ (closer to 0) drives the convergence faster, but also de-emphasizes the relevance of the hyperlink graph structure. This is hardly surprising: if you view the webpage importance scores more like random objects, it is easier to compute the equilibrium. Usually $\theta = 0.85$ is believed to be a pretty good choice. This leads to convergence in tens of iterations while still giving most of the weight to the actual hyperlink graph structure (rather than the randomization component in $\mathbf{G}$).

3.4.2 PageRank as the solution to a linear equation

PageRank sounds similar to distributed power control in Chapter 1. Both of them apply the power method to solve a system of linear equations. The solutions to those equations capture the right engineering configuration in the network, whether that is the relative importance of webpages in a hyperlink graph or the best transmit power vector in a wireless interference environment. This conceptual connection can be sharpened to an exact, formal parallel.

First, we can rewrite the characterization of π^* as the solution to the following linear equation, rather than as the dominant left eigenvector of matrix $\mathbf{G}$ (3.4) that has been our viewpoint so far:

$$(\mathbf{I} - \theta\mathbf{H})^T\pi = \mathbf{v}. \tag{3.6}$$

We will soon prove that (3.6) is true. We can now compare (3.6) with the characterization of the optimal power vector in the distributed power control algorithm in Chapter 1:

$$(\mathbf{I} - \mathbf{DF})\mathbf{p} = \mathbf{v}.$$

Of course, the vectors $\mathbf{v}$ are defined differently in these two cases: in terms of webpage viewing behavior in PageRank and receiver noise in power control. But we see a striking parallel: the self-consistent importance-score vector π and the optimal transmit-power vector $\mathbf{p}$ are both solutions to a linear equation with the following structure: identity matrix minus a scaled version of the network connectivity matrix.

In PageRank, the network connectivity is represented by the hyperlink matrix $\mathbf{H}$. This makes sense since the key factor here is the hyperlink connectivity pattern among the webpages. In power control, the network connectivity is represented by the normalized channel gain matrix $\mathbf{F}$. This makes sense since the key factor here is the strength of the interference channels.

In PageRank, the scaling is done by the single scalar θ. In power control, the scaling is done by many scalars in the diagonal matrix $\mathbf{D}$: the target SIR for each user. To make the parallel exact, we may think of a generalization of the Google matrix $\mathbf{G}$ where each webpage has its own scaling factor θ.

The general theme for solving these two linear equations can be stated as follows. Suppose you want to solve a system of linear equations $\mathbf{Ax} = \mathbf{b}$, but do not want to directly invert the square matrix $\mathbf{A}$. You might be able to split $\mathbf{A} = \mathbf{M} - \mathbf{N}$, where matrix $\mathbf{M}$ is invertible and its inverse $\mathbf{M}^{-1}$ can be much more easily computed than $\mathbf{A}^{-1}$.

The following **linear stationary iteration** ("linear" because the operations are all linear, and "stationary" because the matrices themselves do not vary over iterations) over times indexed by k:

$$\mathbf{x}[k] = \mathbf{M}^{-1}\mathbf{N}\mathbf{x}[k-1] + \mathbf{M}^{-1}\mathbf{b}$$

will converge to the desired solution:

$$\lim_{k\to\infty} \mathbf{x}[k] = \mathbf{A}^{-1}\mathbf{b},$$

from any initialization $\mathbf{x}[0]$, provided that the largest eigenvalue of $\mathbf{M^{-1}N}$ is smaller than 1. Both DPC and PageRank are special cases of this general algorithm.

But we still need to show that (3.6) is indeed equivalent to (3.4): a π that solves (3.6) also solves (3.4), and vice versa. First, starting with a π that solves (3.6), we can easily show the following string of equalities:

$$\begin{aligned}\mathbf{1}^T\mathbf{v} &= \mathbf{1}^T(\mathbf{I} - \theta\mathbf{H})^T\pi \\ &= \mathbf{1}^T\pi - \theta(\mathbf{H1})^T\pi \\ &= \mathbf{1}^T\pi - \theta(\mathbf{1} - \mathbf{w})^T\pi \\ &= \pi^T(\theta\mathbf{w} + (1-\theta)\mathbf{1}),\end{aligned}$$

where the first equality uses (3.6) and the third equality uses the fact that summing each row of $\mathbf{H}$ gives a vector of 1s (except those rows corresponding to dangling webpages). The other two equalities are based on simple algebraic manipulations.

But $\mathbf{1}^T\mathbf{v} = 1$ by design, so we know

$$\pi^T(\theta\mathbf{w} + (1-\theta)\mathbf{1}) = 1.$$

Now we can readily check that $\pi^T\mathbf{G}$, using its definition in (3.5) and the above equation, equals $\theta\pi^T\mathbf{H} + \mathbf{v}^T$.

Finally, using one more time the assumption that π satisfies (3.6), i.e., $\mathbf{v} = (\mathbf{I} - \theta\mathbf{H})^T\pi$, we complete the argument:

$$\pi^T\mathbf{G} = \theta\pi^T\mathbf{H} + \pi^T(\mathbf{I} - \theta\mathbf{H}) = \theta\pi^T\mathbf{H} - \theta\pi^T\mathbf{H} + \pi^T = \pi^T.$$

Therefore, any π solving the linear equation (3.6) is also a dominant left eigenvector of $\mathbf{G}$ that solves (3.4).

Vice versa, a π that solves (3.4) also solves (3.6), which can be similarly shown.

3.4.3 Scaling up and speeding up

It is not easy to adjust the parameters in PageRank computation. We discussed the role of θ before, and we know that, when θ is close to 1, PageRank results become very sensitive to small changes in θ, since the importance matrix $(\mathbf{I} - \theta\mathbf{H})^{-1}$ becomes very big.

There is also substantial research going into designing the right randomization vector $\mathbf{v}$, e.g., the entries of the $\mathbf{H}$ matrix: web surfers likely will not pick all of the hyperlinked webpages equally likely, and their actual behavior can be recorded to adjust the entries of $\mathbf{H}$.

The biggest challenge to running PageRank, however, is *scale*: how does Google scale up PageRank to really large matrices? How can we quickly compute and update the rankings? There are both storage and computational challenges. Many interesting solutions have been developed over the years, including the following five. The first one is a computational acceleration method. The other four are *approximations*, two of which change the notion of optimality and two of which restructure the graph of hyperlinked webpages.

1. *Decompose* $\mathbf{H}$. A standard triangular decomposition gives $\mathbf{H} = \mathbf{DL}$, where $\mathbf{D}$ is a diagonal matrix with entries being $1/O_i$, and $\mathbf{L}$ is a binary adjacency matrix. So only integers, instead of real numbers, need to be stored to describe $\mathbf{H}$. Suppose there are N webpages, and, on average, each webpage points to M webpages. N is huge; maybe in the billions, and M is very small; often 10 or less. Instead of NM multiplications, this matrix decomposition reduces it to just M multiplications.
2. *Relax the meaning of convergence.* It is not the values of importance scores that matter to most people, it is just the *order* of the webpages, especially the top ones. So once the computation of PageRank is sure about the order, there is no need to further improve the accuracy of computing π towards convergence.
3. *Differentiate among the webpages.* The PageRank algorithm as applied to most webpages quickly converges, and can be locked while the other webpages' PageRanks are refined. This is an approximation that works particularly well when the importance scores follow the power law that we will discuss in Chapter 10.
4. *Leave the dangling nodes out.* There are many dangling nodes out there, and their behaviors in the matrices and the computation involved are pretty similar. So they might be grouped together and not be updated in order to speed up the computation.
5. *Aggregation of webpages.* When many nodes are lumped together into a cluster, then *hierarchical* computation of PageRanks can be recursively computed, first treating each cluster as one webpage, then distributing the PageRank of that cluster among the actual webpages within that cluster. We will visit an example of this important principle of building hierarchy to reduce the computation (or communication) load in a homework problem.

3.4.4 Beyond the basic search

There is another player in the game of search: companies that specialize in increasing a webpage's importance scores, possibly pushing it into the top few search results, or even to the very top spot. This industry is called **Search Engine Optimization** (SEO). There are many proprietary techniques used by SEO companies. Some techniques enhance content-relevance scores, sometimes by adding bogus tags in the html files. Other techniques increase the importance score, sometimes by adding links pointing to a customer's site, and sometimes by creating several truly important webpages and then attaching many other webpages as their outgoing neighbors.

Google is also playing this game by detecting SEO techniques and then updating its ranking algorithm so that the artificial help from SEO techniques is minimized. For example, in early 2011, Google made a major update to its ranking algorithm to counter the SEO effects, and another significant update in May 2012 to further reduce the benefit of having not-so-important webpages pointing to a given webpage.

There are also many useful variants to the basic type of search we discussed, e.g., personalized search based on user's feedback on how useful she finds the top webpages in the search result. Multimedia search is another challenging area: searching through images, audios, and video clips requires very different ways of indexing, storing, and ranking the content than text-based search.

Summary

Box 3 PageRank computes webpages' importance scores

We can view the hyperlinked webpages as a network, and use the connectivity patterns to rank the webpages by their importance scores. If many important webpages point to a given webpage, that webpage maybe also important. PageRank uniquely defines and efficiently computes a consistent set of importance scores. This set of scores can be viewed as the dominant eigenvector of a Google matrix that consists of a network connectivity part and a randomization part.

Further Reading

The PageRank algorithm is covered in almost every single book on network science these days. Some particularly useful references are as follows.

1. Back in 1998, the Google founders wrote the following paper explaining the PageRank algorithm:

S. Brin and L. Page, "The anatomy of a large-scale hypertextual Web search engine," *Computer Networks and ISDN Systems*, vol. 33, pp. 107–117, 1998.

2. The standard reference book devoted to PageRank is

A. N. Langville and C. D. Meyer, *Google's PageRank and Beyond*, Princeton University Press, 2006.

3. The following excellent textbook on the computer science and economics of networks has a detailed discussion of navigation on the web:

D. Easley and J. Kleinberg, *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*, Cambridge University Press, 2010.

4. Dealing with non-negative matrices and creating linear stationary iterations are well documented, e.g., in the following reference book:

A. Berman and R. J. Plemmons, *Nonnegative Matrices in the Mathematical Sciences*, Academic Press, 1979.

5. Computational issues in matrix multiplication are treated in textbooks like this one:

G. Golub and C. F. van Van Loan, *Matrix Computations*, 3rd edn., The Johns Hopkins University Press, 1996.

Problems

3.1 *PageRank sink* ⋆

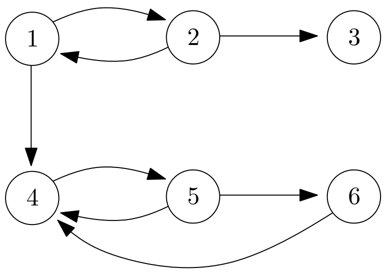

Figure 3.5 A simple network of webpages with a sink node.

Write out the **H** matrix of the graph in Figure 3.5. Iterate $\pi[k]^T = \pi[k-1]^T\mathbf{H}$, where $k = 0, 1, 2, \ldots$, and let the initialization be

$$\pi[0] = \begin{bmatrix}1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6\end{bmatrix}^T.$$

What problem do you observe with the converged π^* vector?

3.2 *Cyclic ranking* ⋆

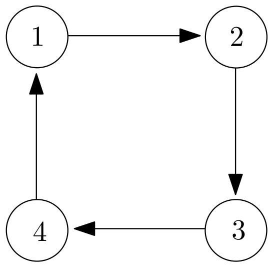

Figure 3.6 A simple network of webpages with a cycle.

Write out the **H** matrix of the graph in Figure 3.6. Iterate $\pi[k]^T = \pi[k-1]^T\mathbf{H}$, where $k = 0, 1, 2, \ldots$, and let the initialization be

$$\pi[0] = \begin{bmatrix}1/2 & 1/2 & 0 & 0\end{bmatrix}^T.$$

What happen to the vectors $\{\pi[k]\}$ as k becomes large? Solve for π^* such that $\pi^{*T} = \pi^{*T}\mathbf{H}$ and $\sum_i \pi_i^* = 1$.

3.3 *PageRank with different θ* ⋆⋆

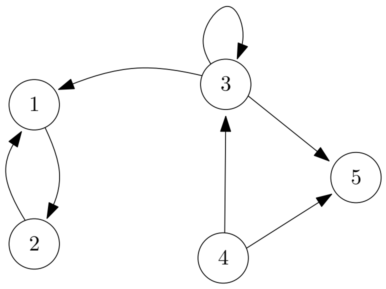

Figure 3.7 A simple example to try page PageRank with different θ.

Compute the PageRank vector π^* of the graph in Figure 3.7, for $\theta = 0.1, 0.3, 0.5$, and 0.85. What do you observe?

3.4 *Block aggregation in PageRank* ⋆⋆

Set $\theta = 0.85$ and start with any normalized initial vector $\pi[0]$.

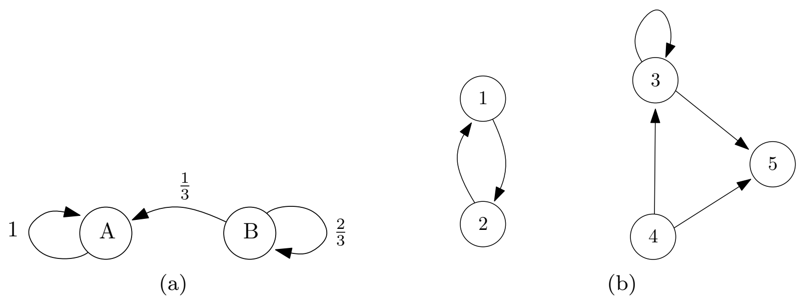

Figure 3.8 An example of hierarchical PageRank. Two different graphs that will be superimposed later.

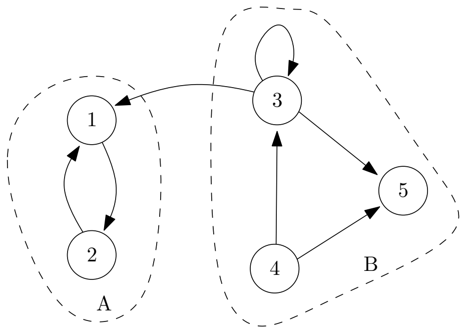

Figure 3.9 Subgraphs A and B in Figures 3.8(a) and 3.8(b) are composed into a single graph.

(a) Compute the PageRank vector $\begin{bmatrix}\pi_A^* & \pi_B^*\end{bmatrix}^T$ of the graph in Figure 3.8(a) with

$$\mathbf{H} = \begin{bmatrix} 1 & 0 \\ 1/3 & 2/3 \end{bmatrix}.$$

Note the uneven splitting of link weights from node B. This will be useful later in the problem.

(b) Compute the PageRank vectors $\begin{bmatrix}\pi_1^* & \pi_2^*\end{bmatrix}^T$ and $\begin{bmatrix}\pi_3^* & \pi_4^* & \pi_5^*\end{bmatrix}^T$ of the two graphs in Figure 3.8(b).

(c) If we divide the graph in Figure 3.7 into two blocks as shown in Figure 3.9, we can approximate π^* in the previous question by

$$\tilde{\pi}^* = \begin{bmatrix}\pi_A^* \cdot \begin{bmatrix}\pi_1^* & \pi_2^*\end{bmatrix} & \pi_B^* \cdot \begin{bmatrix}\pi_3^* & \pi_4^* & \pi_5^*\end{bmatrix}\end{bmatrix}^T.$$

Compute this vector. Explain the advantage, in terms of computational load, of using this approximation instead of directly computing π^*.

3.5 *Personalized ranking (open-ended)*

How would you solicit and aggregate feedback from individual users to enable *personalized* ranking?