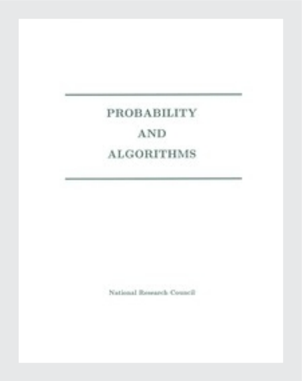

PROBABILITY
AND
ALGORITHMS

National Research Council

Probability and Algorithms

DETAILS

188 pages | 8.5 x 11 | PAPERBACK
ISBN 978-0-309-04776-0 | DOI 10.17226/2026

AUTHORS

Panel on Probability and Algorithms, National Research Council

PROBABILITY AND ALGORITHMS

Panel on Probability and Algorithms
Committee on Applied and Theoretical Statistics
Board on Mathematical Sciences
Commission on Physical Sciences, Mathematics, and Applications
National Research Council

National Academy Press
Washington, D.C. 1992

NOTICE: The project that is the subject of this report was approved by the Governing Board of the National Research Council, whose members are drawn from the councils of the National Academy of Sciences, the National Academy of Engineering, and the Institute of Medicine. The members of the committee responsible for the report were chosen for their special competences and with regard for appropriate balance.

This report has been reviewed by a group other than the authors according to procedures approved by a Report Review Committee consisting of members of the National Academy of Sciences, the National Academy of Engineering, and the Institute of Medicine.

The National Academy of Sciences is a private, nonprofit, self-perpetuating society of distinguished scholars engaged in scientific and engineering research, dedicated to the furtherance of science and technology and to their use for the general welfare. Upon the authority of the charter granted to it by the Congress in 1863, the Academy has a mandate that requires it to advise the federal government on scientific and technical matters. Dr. Frank Press is president of the National Academy of Sciences.

The National Academy of Engineering was established in 1964, under the charter of the National Academy of Sciences, as a parallel organization of outstanding engineers. It is autonomous in its administration and in the selection of its members, sharing with the National Academy of Sciences the responsibility for advising the federal government. The National Academy of Engineering also sponsors engineering programs aimed at meeting national needs, encourages education and research, and recognizes the superior achievements of engineers. Dr. Robert M. White is president of the National Academy of Engineering.

The Institute of Medicine was established in 1970 by the National Academy of Sciences to secure the services of eminent members of appropriate professions in the examination of policy matters pertaining to the health of the public. The Institute acts under the responsibility given to the National Academy of Sciences by its congressional charter to be an adviser to the federal government and, upon its own initiative, to identify issues of medical care, research, and education. Dr. Kenneth I. Shine is president of the Institute of Medicine.

The National Research Council was organized by the National Academy of Sciences in 1916 to associate the broad community of science and technology with the Academy's purposes of furthering knowledge and advising the federal government. Functioning in accordance with general policies determined by the Academy, the Council has become the principal operating agency of both the National Academy of Sciences and the National Academy of Engineering in providing services to the government, the public, and the scientific and engineering communities. The Council is administered jointly by both Academies and the Institute of Medicine. Dr. Frank Press and Dr. Robert M. White are chairman and vice chairman, respectively, of the National Research Council.

The National Research Council established the Board on Mathematical Sciences in 1984. The objectives of the board are to maintain awareness and active concern for the health of the mathematical sciences and to serve as the focal point in the National Research Council for issues connected with the mathematical sciences. The board conducts studies for federal agencies and others and maintains liaison with the mathematical sciences communities, academia, professional societies, and industry.

Support for this project was provided by the Air Force Office of Scientific Research, the Army Research Office, and the National Science Foundation.

Library of Congress Catalog Card No. 92-60700

International Standard Book Number 0-309-04776-5

Additional copies of this report are available from: National Academy Press 2101 Constitution Avenue, NW Washington, DC 20418

S-612

Printed in the United States of America

PANEL ON PROBABILITY AND ALOGRITHMS

J. MICHAEL STEELE, University of Pennsylvania, *Chair*
DAVID ALDOUS, University of California at Berkeley
DIMITRIS J. BERTSIMAS, Massachusetts Institute of Technology
EDWARD G. COFFMAN, AT&T Bell Laboratories
DORIT HOCHBAUM, University of California at Berkeley
MICHA HOFRI, University of Houston
JEFFREY C. LAGARIAS, AT&T Bell Laboratories
SCOTT T. WEIDMAN, Senior Staff Officer

COMMITTEE ON APPLIED AND THEORETICAL STATISTICS

WILLIAM F. EDDY, Carnegie Mellon University, *Chair*
YVONNE BISHOP, U.S. Department of Energy
DONALD P. GAVER, Naval Postgraduate School
PREM K. GOEL, Ohio State University
DOUGLAS M. HAWKINS, University of Minnesota
DAVID G. HOEL, National Institute of Environmental Health Sciences
JON KETTENRING, Bellcore
CARL N. MORRIS, Harvard University
KARL E. PEACE, Biopharmaceutical Research Consultants
JAYARAM SETHURAMAN, Florida State University
JOHN R. TUCKER, Staff Officer

BOARD ON MATHEMATICAL SCIENCES

Ex Officio Member

Staff

COMMISSION ON PHYSICAL SCIENCES, MATHEMATICS, AND APPLICATIONS

Preface

The Panel on Probability and Algorithms was constituted by the National Research Council in 1991 and charged with writing a report surveying both the topic of probabilistic algorithms, where randomization is a part of the internal calculation, and the probabilistic analysis of algorithms, in which one uses a probability model to deepen the understanding of how an algorithm functions in practice. Probabilistic algorithms—simulated annealing is one example—incorporate randomness into the underlying logic. For problems with huge solution spaces, or those where deterministic models of the processes involved do not exist, such algorithms have been very exciting developments. Probabilistic algorithms can solve certain problems faster than is possible by any deterministic algorithm. The probabilistic analysis of algorithms is a refinement of worst-case analysis, which is often too pessimistic compared to the performance of algorithms in actual practice.

Both uses of probability show tremendous promise, yet the research opportunities outnumber the people available to explore them. The panel hopes that more researchers, especially young researchers, will be intrigued by the possibilities and attracted to this field.

The panel gratefully acknowledges the substantial contributions of its contributing authors, Joan Feigenbaum, David Johnson, George Lueker, Bruce Maggs, Vijaya Rarnachandran, Ron Shamir, Peter Shor, and John Tsitsiklis. These colleagues greatly strengthened the report with their expertise. The contributions of Hans Oser, who served for a time as the project staff officer, are also appreciated.

Contents

1. Introduction

J. Michael Steele,[1] University of Pennsylvania and David Aldous, University of California at Berkeley

The theory of algorithms has undergone an extraordinarily vigorous development over the last 20 years, and probability theory has emerged as one of its most vital partners. University courses, research monographs, and specialized journals have been developed to serve this partnership, yet there is still a need to communicate the progress in this area to the wider audience of computer scientists, mathematicians, and individuals who have a stake in the contributions that mathematical research can make to technology.

The main purpose of this report is to give an understanding of the power that comes from applying probability in the theory of algorithms, but an equally essential aim is to point out the *variety* of ways in which probability plays a role. One useful step in understanding this variety comes from making a clear distinction between the subject of *probabilistic algorithms* and the subject of *probabilistic analysis of algorithms*. Confusion sometimes arises over what methods are properly called "probabilistic (or randomized) algorithms"—and indeed there are some gray areas. Still, if one does not press for too fine a point, there are considerable organizational and conceptual benefits in drawing the distinction between probabilistic algorithms and the probabilistic analysis of a (possibly deterministic) algorithm.

One common distinction is that probabilistic algorithms, unlike deterministic ones, make random choices when computing. They are commonly referred to as "coin-flipping algorithms." Such algorithms will produce (possibly) different results for the same problem when posed in different circumstances. On the other hand, the probabilistic analysis of an algorithm

[1] J. Michael Steele's research was supported in part by NSF grant DMS88-12868, AFOSR grant 89-0301, ARO grant DAAL03-89-G-0092, and NSA grant MDA-904-H-2034.

incorporates randomness into the *data* processed by an algorithm. Properly speaking, we are considering the pair (*algorithm, problem instance*) and probabilistically exploring the algorithm behavior over a large variety of problem instances. Typically, the analyst can make statements about the probability of selecting a particular instance, or focus attention on the distribution of suitable variables that describe the problem instance. The task is then to relate the algorithm performance to these variables.

This introductory chapter provides several simple illustrations of the distinction between the design of probabilistic algorithms and the probabilistic analysis of algorithms. An excellent examples-oriented survey of probabilistic algorithms is that of Karp (1991).

1.1 PROBABILISTIC ALGORITHMS

1.1.1 Everyday Examples

Before developing more serious examples that require detailed mathematical description, it seems useful to provide a couple of everyday analogies. Mathematics often speaks best for itself, and one should not read too much into analogies, but because there are important uses of probabilistic algorithms that can be explained without any technical prerequisites, it seems appropriate to look at them early on.

We have all had the experience of walking along a narrow path and encountering someone who is approaching on the same side of the path. As the individuals draw closer, one moves to the other side of the path in order to let the other person pass, but often the other person does the same simultaneously. This little shuffle continues one or two more times before the two people end up on opposite sides of the path and can pass each other.

Although people resolve such difficulties with little more cost than a moment's embarrassment, the analogous situation is more serious when packets of information end up vying for the use of the same place at the same time in a communication channel. When one tries to program machines to avoid such deadlocks, there are decided drawbacks to most deterministic rules. The dogged consistency of machines is such that the shuffling from side to side can go on unabated until an outside action halts the dance. This is an unacceptable state of affairs that one ought to be able to avoid through thoughtful design.

One theme of this report is that a natural course of action when one is faced by the disadvantages of purely deterministic rules is to introduce some sort of randomness into the protocol. Each process could continue to follow its own **randomized** rule without regard for the rule being followed by the other process, and the eternal dance would be avoided. This simple idea is at the heart of many of the communication protocols in use in the world today.

The larger idea onto which this flavor of application can be attached might be called "randomization to avoid special coincidences." Mathematicians can find analogs of this phenomenon in many other fields, and, as more technical examples will illustrate later, there are more profound consequences to the idea than are clone justice to by this first quick, everyday example.

Our second everyday example involves opinion polls. These are so widely discussed and so well understood that it is not easy to evoke a proper awareness that there is actually quite a bit of magic in the technology—enough so that there is still cutting-edge research in computer science related to the subject. Still, in order not to get ahead of our story, let us first consider a question that is typical of the sort addressed by Gallup and other famous pollsters: What percentage of the U.S. voting population feels that the recent pay raise voted for itself by the U.S. Senate was well deserved? A fact that was completely unavailable to the founding fathers, but which is much a part of modern political life, is that one can obtain a useful answer to the question at a cost that is a small fraction of the cost of conducting a census of the whole voting populace.

One point that deserves to be underscored in this example is that the precise question that has been posed has nothing a priori to do with probability. If we are totally faithful to the precise phrasing of the question, we have to agree that it is one that can be answered only by a complete census as long as one insists on an *exact* answer. Still, a useful answer need not be perfectly exact, and the full complement of political insight is likely to be gained by knowing the answer to within plus or minus two percentage points. This fact is easily seen and accepted in this context, yet when one reviews the theory of algorithms one finds that tremendous effort is often expended to get exact answers where approximate ones might serve just as well. Thus, willingness to accept approximation seems to be one door for probability to enter. More surprising mathematical examples to be considered below will show that this is not the only door.

Still, in the context of our polling question, an approximation dearly serves the policy purpose just as well as a complete census. If one selects a random sample of about 2,500 from the target population and asks, "Do you agree that the pay raise that the U.S. Senate gave itself is well deserved?", the percentage of individuals who say yes provides an estimate of the percentage of the whole populace that would answer in the same way. It is a consequence of elementary probability theory that our estimate will be correct to within ±2 percent on better than 95 percent of the occasions on which such a sampling experiment is conducted.

This is a familiar tale, even if told a bit more precisely than usual, and the reader may legitimately ask why this old tale is being told here. The retelling is not occasioned to celebrate polling, which certainly draws enough of its own attention. The point is not even to recall the often missed subtlety that the required sample size depends only on the precision required, and not on the size of the population about which the inference is being made; though it is interesting that the senators from Rhode Island find it as costly to obtain information about the voting populace of that state as it is for the National Committee to get the corresponding information about the nation as whole.

The point is rather to see that the sampling techniques of political pollsters are actually probabilistic algorithms; one uses exogenous randomization to obtain an approximation to a problem that would be prohibitively expensive to answer exactly. The key point for us is not the approximation issue. Approximation is a common but not inevitable feature of probabilistic algorithms. The key point is that one pours in randomness that was not there to begin with. In the pollster's case the randomness that was added was that used in making the random selection. What was purchased at the price of this extra complication was the applicability of the probability theory that enabled us to quantify the quality of our solution. These are features that one finds throughout the theory of probabilistic algorithms.

To be sure, there are differences in flavor between the pollsters' techniques and those applied in computer science. The pollsters' methods are applied in a social rather than a computational context, and, perhaps as a result, the process looks unsophisticated, or even straightforward. Individuals will sometimes disagree, but it seems useful to hold that the pollsters' techniques are just as much a probabilistic algorithm as any being studied in computer science. The difference of context does not alter the logical structure, and the perceived difference in sophistication comes in good measure from the fact that our example did not go into any subtleties such as how one might really draw the sample and whether some trickier sampling scheme might do better than a uniform random sample.

This pollster problem completes the second of our two everyday exam ples. Both were illustrations of randomization applied in algorithms. In one case the randomization offered escape from the "special situation" traps into which deterministic processes can fall. In the second case, we gained great savings of cost at the price of accepting an approximation, and we were able to quantify the quality of our approximation because we were able to add just the kind of extra randomness to which a compelling theory could be applied.

1.1.2 Hashing

Some of the earliest examples of probabilistic algorithms were invented in the context of storage algorithms, and one of most influential of the ideas to be developed to speed up the access of stored information is **hashing**. This device is used in many important computational contexts and even lives at the heart of some computer operating systems. Still, it can be described in the style of an everyday example.

Consider an office where 50 employees receive mail. The thoughtful employer typically provides a set of somewhat more than 50 mailboxes so each employee can be assigned one. Conceivably, the assignment of physical mailboxes to individuals could be done in lots of different ways. But the conventional method is so familiar that most people would call it obvious. The tradition is to physically label mailboxes with employees' names, in alphabetical order (this suggests an ordering of the boxes, usually left-to-right within rows and top-to-bottom across rows, but we digress). To see that this arrangement is not the only conceivable one, consider instead a hotel that has 50 rooms and receives letters addressed to its guests. The hotel could use the arrangement above, but that would be silly because guests arrive and leave so rapidly that the staff would waste time moving name labels. Instead, hotels label their mailboxes by room number, and when letters arrive they look up the addressee's room number. Hashing abstracts the hotel's procedure. Suppose we have *n* storage locations numbered 1, 2, . . . , *n* and wish to store $m \leq n$ items, where each item has (say) a name in plain English. We specify some **hash function** $f : \{\text{all names}\} \rightarrow \{1, 2, \ldots, n\}$. For each item in turn we compute *f*(name) = *i*, say, and attempt to store the item in location *i*. We then need a rule to tell us what to do if location *i* is already occupied; the simplest rule, **hashing with linear probing**, is to look at locations *i* + 1, *i* + 2, ... until an empty location is found and store the item there. Note that once all the items are stored, they are simple to locate. Given the name, compute *f*(name) = *i*, say, and look at locations *i, i* + 1, ... until the item is found or an empty location is found, in which case the item cannot be present.

This algorithm is appropriate if storage is cheap and we desire to locate items very quickly. Under plausible assumptions, the mean number of locations checked in searching for an item (averaged over items) is approximately a function of the density *m/n* only; that is to say, it does not depend on the total number of items to be stored. The conceptual idea is to choose a hash function that "scrambles" the name so that the resulting value can be regarded as random and uniform on 1, 2, . . . , *n*. Thus, the mathematical analysis of the performance of hashing with linear probing can be clone under the assumption that the hashed values of the *m* items are independent uniform random variables. To justify this assumption for a particular hash function would consequently involve the same issues as justifying a pseudorandom number generator.

Though it is not difficult to analyze the scheme just described (see Knuth, 1973, §6.4), the main features of this scheme can be seen from a trivial-to-analyze but less practical variant in which one specifies a sequence of hash functions and resolves collisions by applying the next hash function. Here, after inserting the (*i*+1)st item, one will need to search a mean number $\frac{n}{n-i}$ of locations until finding an empty location. The same number is needed to search for the item, so the mean search length (over all *m* items) is

$$\frac{1}{m}\sum_{i=1}^{m}\frac{n}{n-i+1} \sim \frac{n}{m}\log\left(\frac{1}{1-\frac{n}{m}}\right) .$$

The linear probing scheme turns out to require more searching because the occupied locations tend to cluster, and its analysis is more taxing. Still, both linear probing and multiple hashing share the fundamental qualitative feature of depending only on the ratio *m/n*.

1.1.3 Geometry

One of the areas in which probabilistic algorithms have recently proved to be exceptionally powerful is computational geometry. The body of this report does not pursue these applications except somewhat in passing; the reader is referred to Seidel (1992), which recounts several applications and gives references to further work. To give some sense of the way that randomization can come into play in computational geometry, we look briefly at one of the examples treated by Seidel.

The **Delaunay triangulation** of a set S of n points in the plane is the graph with vertex set S that puts an edge between points x and y if and only if there is a disc with x and y on its boundary that contains no other points of S. Algorithms for computing the Delaunay triangulation have been well studied. There is a known deterministic algorithm for computing the tessellation in $O(n \log n)$ steps, and in the worst case this order cannot be improved.

For the special case where S consists of the vertices of a convex polygon, a simple probabilistic algorithm is available. The key idea is to let s_n be one of the points in S and let $S_{n-1} = S \setminus \{s_n\}$. In this special case S_{n-1} still forms the vertices of a convex polygon. Suppose we are given the tessellation of S_{n-1} and want to compute the tessellation of S. There is an algorithm for doing this (see Seidel, 1992) which we shall not describe here: if we are lucky, we may only have to add two edges to s_n, but in general we must delete some existing edges and retriangulate.

Obviously, given such an algorithm we can order the points of S arbitrarily as $s_1, s_2, \ldots, S_n$ and apply the algorithm to update the tessellation as each new point is added. The efficiency of this process will clearly depend on the order of the s_i, but it is not easy to envisage an optimal order. The key idea is to proceed as in Quicksort and put S in random order. Then the final update can be rephrased as follows: pick s_n at random from S, suppose we have constructed the triangulation on $S \setminus \{s_n\}$, and apply the update algorithm. A geometrical argument shows that, for each choice of s_n, the number of steps needed is on the order of the degree of s_n in the tessellation on S. But it is easy to see that the average degree of the vertices in a Delaunay tessellation is less than 4, so because s_n is random we conclude that the expected number of steps in the final update is O (1), and hence the total number of steps is $O(n)$.

1.1.4 Competitive Analysis

Traditionally, algorithms have been compared using **worst-case** analysis or **average-case** analysis, in other words by considering the worst possible input or by putting some probability measure (perhaps chosen for mathematical simplicity rather than realism) on inputs. Recently, attention has been paid to a different mode of analysis called **competitive analysis**. In particular, this has been applied to on-line algorithms (defined in Chapter 7). The idea is to compare a given on-line algorithm with the optimal off-line algorithm and define the **competitiveness coefficient** to be the smallest *c* such that, for some *b*,

$$\begin{aligned}&\{\textit{Cost using on-line algorithm}\}\\ &\quad \leq c \times \{\textit{Cost using optimal algorithm}\} + b,\end{aligned}$$

for all inputs. One specific problem where this method of analysis has been used is the **caching problem**.[2] Imagine a cache (fast memory) that can hold *n* items and a slower external memory that we can model as having unlimited capacity. A requested item that is not in the cache must be retrieved from the main memory at unit cost. To use these memories efficiently, an algorithm is needed to specify when items are switched between cache and main memory. Fiat et al. (1991) invented a simple probabilistic algorithm, the **marker algorithm**. Initially, let none of the *n* items in the cache be marked. If an item in the cache is requested, then it is marked. If an item outside the cache is requested, then an unmarked item in the cache is chosen uniformly at random and switched with the requested item, which is then marked. Eventually, all items in the cache are marked, at which time remove all marks and start again. Fiat et al. proved this algorithm has competitiveness coefficient at most $2\sum_{i=1}^{n} 1/i \sim 2\log n$. Note the simplicity of the randomization: it is hard to imagine a simple deterministic algorithm performing so well.

1.1.5 Random Constructions

A surprising connection between mathematical probability and the theory of algorithms is the idea of proving mathematical results about random objects by studying the behavior of an algorithm for constructing the random object. We illustrate this with the classical topic of random permutations. Suppose we want to generate, inductively on *n*, the uniform random permutation of *n* objects into *n* positions. Picturing the objects as physical files in a file cabinet, the natural update algorithm for adding the *n*th object is: pick *p* uniformly on 1, . . . , *p*, move objects at positions *i* = *p, p*+1, . . . , *n* to positions *i* + 1, and put object *n* at position *p*. In a computer the natural update algorithm is: put object *n* at position *n*, pick *p* uniformly on 1, . . . , *p*, and switch the objects at positions *n* and *p*.

Though the latter algorithm is essentially the standard optimal algorithm for computer simulation of a random permutation, here is a more

[2] Our treatment follows Raghavan (1990).

complicated algorithm that turns out to be useful for mathematical purposes:

The Chinese restaurant process. Mathematicians at a conference agree to dine at a Chinese restaurant.[3] They arrive separately, and so the *i*th arrival has *i* choices of where to sit: next (clockwise, say) to one of the *i*-1 mathematicians already present, or at an unoccupied table. Suppose the choice is random and uniform. After *n* arrivals, the pattern of mathematicians (labeled by arrival order) at tables can be regarded as the cycle representation of a permutation, and it is easy to see that the sequential uniform random choices create a uniform random permutation.

Suppose we want to study the random number C_n of cycles in a random permutation. In terms of the Chinese restaurant algorithm, C_n is just the number of occupied tables. But the *i*th arrival has chance 1/*i* of starting a new table, so without calculation we can see that the expectation of C_n is $\sum_{i=1}^{n} 1/i$. (Classical probability theory gives a normal limit distribution.)

1.1.6 Testing Equality and Faith

Suppose one has a polynomial in *n* variables $p\,(x_1, x_2, \ldots, x_n)$ that represents a possible identity. Naturally, it is desirable to know if the identity is valid for all $x_1, x_2, \ldots, x_n$. For example, suppose one did not know about the modulus identity for quaternions but had somehow come to suspect that, for $\alpha, \beta, \gamma, \delta, \alpha', \beta', \gamma', \delta' \in \mathbb{R}$,

$$\begin{aligned}(\alpha^2+\beta^2+\gamma^2+\delta^2)(\alpha'^2+\beta'^2+\gamma'^2+\delta'^2) = \\ (\alpha\alpha'-\beta\beta'-\gamma\gamma'-\delta\delta')^2+(\alpha\beta'+\beta\alpha'+\gamma\delta'-\delta\gamma')^2 \\ +(\alpha\gamma'+\gamma\alpha'+\delta\beta'-\beta\delta')^2+(\alpha\delta'+\delta\alpha'+\beta\gamma'-\gamma\beta')^2\end{aligned}$$

Is there a quick and painless way to decide if this conjectured identity is indeed valid for all values of the arguments?

There is way to answer this question that suggests itself rather naturally from the probabilistic point of view and that also serves to point out some subtleties in the algorithmic uses of probability. The idea can also be used to test one's faith in probability theory.

Because nonzero polynomials in *n* variables cannot vanish on a set of positive measure, the brave probabilist can check identity almost trivially.

[3] Chinese restaurants often have large circular tables.

All one has to do is choose a point (α, β, γ, δ, α', β', γ', δ') at random from $[0, 1]^8$ to see if the point satisfies our would-be identity. If the expression in question is not an identity, the two sides will evaluate to different values with probability one. Thus, without any call on sampling theory or repeated trials, one can answer the point in question with probabilistic certainty and very little arithmetic.

Still, the answer is too easy not to provoke doubts. Our ability to choose numbers at random from sets like $[0, 1]^8$ comes belatedly into question, and we are reminded that computers do not actually operate on real numbers. There is clearly some need to discretize the probabilists' suggestion, and there are important details to be resolved. Our first help is to be found in a discrete version of our statement that a nontrivial polynomial in n variables cannot vanish on a set of positive measure in $\mathbf{R}^n$.

Lemma 1 *Suppose p* $(x_1, x_2, \ldots, x_n)$ *is a polynomial of degree d with integer coefficients. If values* X_1, $X_2, \ldots, X_n$ are chosen independently from the uniform distribution on $\{0, 1, 2, \ldots, d\}$, *then*

$$P[p(X_1, X_2, \ldots, X_n) = 0] < 1/2.$$

This lemma can be found in Schwartz (1980), and it can be proved by induction on the number of variables. The main point of the lemma is that it tells us that we can put our relation to the test *k* times, and, if it is not a true identity, then we will discover the fact with probability at least $1\text{-}2^{-k}$. After suffering prior doubts about the possibility of selecting points at random from $[0, 1]^8$, one might worry about how we can make random choices out of $\{0, 1, \ldots, d\}$. Clearly, the situation is better, but still it may not be fully resolved. This question is dealt with at length in Chapter 6.

1.2 PROBABILISTIC ANALYSIS OF ALGORITHMS

1.2.1 Sample Analyses: The Assignment Problem

Probabilistic analysis is inevitably mathematical, so no everyday example can help illustrate this second of the great gates through which probability theory enters the theory of algorithms. Still, for our first such example, we can use a task that often arises as a module in larger computational problems and that is evocatively expressed in language that reflects its origins in industrial engineering. It requires a slight increase in technical level over our everyday examples, but it is still friendly and nontechnical. An additional benefit of the problem is that its analysis offers several surprises that have only recently been discovered.

We suppose that we have a set of n jobs labeled $\{1, 2, \ldots, n\}$ and a set of n machines on which to perform the jobs that are similarly labeled. We further suppose that there is a matrix of numbers c_{ij} that describe the cost of doing job i on machine j. A natural and important computational task is to specify a one-to-one assignment of jobs to machines that minimizes the total cost. In other words, the task is to determine a permutation σ that minimizes

$$A(\sigma) = \sum_{i=1}^{n} c_{i\sigma(i)} \cdot$$

This is called the **assignment problem**, and one can see how it could come about easily on its own or in the context of a larger computational task.

One natural way to try to solve this problem is to use a "greedy" strategy. There are two natural ways to proceed, so we will consider the simplest one first. We look at job 1 and assign it to machine j, where j is chosen to minimize $c1j$ over all $1 \leq j \leq n$. We then successively assign jobs 2, 3, ... , n by choosing at each step we take the least costly among the unassigned machines.

There is a slightly more sophisticated algorithm that takes a more globally greedy perspective. For the first assignment, one chooses the assignment $i \rightarrow j$, where (i, j) are chosen to minimize c_{ij} over all n^2 possible choices. The second assignment then chooses the cheapest possibility from the remaining set of $(n -1)^2$ pairs of feasible assignments. One then continues in this globally greedy way until one has a complete assignment of jobs to machines. The natural problem in this context is to determine which of the two methods is better. Another compelling problem is to determine if there are still other methods that do substantially better than these two greedy procedures.

One way to approach these problems is to assume some probability model on the input data c_{ij} and to calculate appropriate measures of performance under this model. An issue that emerges at this point is that for algorithms that do not return an exact solution, there are two compelling dimensions along which to measure performance: the quality of the solution obtained and the amount of time needed to obtain that solution. For the simple and global greedy algorithms for the assignment problem, the running time analysis does not depend on probability theory, so we will consider it first.

Because one can find the smallest element in a list of k items in time that is bounded by a constant times k, the running time of the first algorithm is bounded by $c[n + (n -1) + (n -2) + ... + 1] = O(n^2)$. The second algorithm requires a little more knowledge to implement efficiently, but for people who have had some exposure to the subject of data structures there are some off-the-shelf tools that come to mind almost immediately. One such tool is a data structure called a **heap**. Given m items that have a total order, one can build this structure in time $O(m)$, and it has the remarkable property that one can delete the largest item in the heap at a cost bounded by $c\log m$ and be left with a new heap that is now of size $m -1$. There are other structures besides heaps that have this property, but unless one is ready to get down to coding, there is no need to do more than note that there is some *abstract* data structure that can be built at the specified cost and that permits deletion at the specified cost.

We can now go about measuring the running time cost of our greedy algorithms. If we put the n^2 costs c_{ij} into a heap at a cost of $O(n^2)$, then even if we have to delete all the items in the heap to build our assignment, the total time used is still only $O(n^2\log n)$. The bottom line of these two running time analyses is that both of the algorithms are quite practical, just as we expected them to be, and the global greedy algorithm is more costly in time to the tune of a factor of log n. This second result is also much as one might have expected.

A bigger surprise comes when one looks at the quality of the solutions that are obtained by the two algorithms. Here, a natural way to proceed is to make some probabilistic assumptions about the cost c_{ij}. One is not likely to find any assumption that is more natural than to take the c_{ij} to be independent, identically distributed random variables. Further, because we are just looking for *some* computable feature of merit that casts light on the quality of the assignments we obtain, we might as well go the rest of the way in making our model easy by assuming that the c_{ij} are uniformly distributed on [0, 1].

This setup does indeed make it easy to compute the expected value of the total assignment cost A_n that is obtained using the simple greedy algorithm. Because the expected value of the minimum of k independent, uniformly distributed random variables is exactly equal to $1/(k + 1)$, we see

$$E(A_n) = 1/(n+1) + 1/n + 1/(n-1) + \ldots + 1/2 \sim \log_e(n) .$$

The analysis of the expected value of A'_n, the cost of the assignment obtained under the global greedy method, is a little bit more difficult to obtain. The problem is that one loses all independence in the model after the very first step. One can nevertheless show by setting up a simple integral equation based on dynamic programming that one again has that $E(A'_n) \sim \log_e(n)$. The punchline is that even though the global greedy algorithm takes a little more time to compute and seems to be a more powerful strategy, the value of the assignment that it yields is typically no better than that one gets by the simple greedy strategy.

Before leaving this example, we note that these heuristics fall short of providing the deepest understanding of $E(A_n^{OPT})$, where $E(A_n^{OPT})$ denotes the least cost of any assignment. Karp (1987) showed by a remarkable argument that $E(A_n^{OPT}) \leq 2$, and the insight provided by Karp's proof has been shown by Dyer et al. (1986) to apply to many other problems that can be expressed as linear programs. Finally, Mézard and Parisi (1987) have offered intriguing calculations based on methods of statistical mechanics that suggest $E(A_n^{OPT}) \sim \pi^2/6$ as $n \to \infty$. For further information on the history of A_n^{OPT} and conjectures concerning its behavior, one can consult Steele (1990).

1.2.2 Quick Lessons from Quicksort

For the second example of a probabilistic analysis of a deterministic algorithm, we have to increase the technical level somewhat. We will first sketch some salient features of an archetypical example: **Quicksort** . This example is a little shopworn, because it is considered in almost every introductory course in the theory of algorithms. Still, Quicksort is an algorithm that has remarkable intrinsic richness, and it is one of the few algorithms to be the subject of an entire monograph (Sedgewick, 1980).

Given a list of n records (R_1, R_2, ... , R_n) that axe associated with n keys (K_1, K_2, ... , K_n) for which there is a total ordering, Quicksort is elegantly expressed as a recursive process that returns an ordered list of the records $(R_{i_1}, R_{i_2}, \ldots, R_{i_n})$ such that $K_{i_1} \leq K_{i_2} \leq \cdots \leq K_{i_n}$.

In its simplest form, Quicksort takes the key K_1 and divides the records into those whose key is less than K_1 and those whose key is greater. The algorithm then recursively calls itself on each of the two resulting sublists. If one resists the temptation to add all of the refinements that help make Quicksort a truly practical procedure, it is a joy to analyze its "average case" behavior.

To sketch the analysis, we first suppose that the list that we begin with is in random order; that is, we suppose that we axe equally likely to begin with the list in any of the *n*! possible orderings. The simplest consequence of this assumption is that the key K_1 is equally likely to have any of the *n* possible ranks, and, moreover, the two sublists produced by splitting at K_1 are again in random order.

If we let Q_n denote the expected number of comparisons that are needed to sort a list of *n* items under the model that the input lists are in random order, then it is easy to check that Q_n satisfies the recursion

$$Q_n = (n-1) + \frac{1}{n}\sum_{k=0}^{n-1}\{Q_k + Q_{n-k-1}\}\,.$$

This equation expresses the fact that the first split requires *n* -1 comparisons, and if *k* is the number of keys less than K_1, one then has to call the Quicksort procedure on two lists of size *k* and *n*-*k*-1, respectively. The only remaining point is that by our randomness assumption, K_1 has probability 1/*n* of having 0, 1, ... , *n* -1 keys less than itself.

To gain any insight about the efficiency of the Quicksort process, one still needs to solve the recursion, or at least to extract the asymptotics of Q_n from it. This cast be done without great difficulty, and one finds that $Q_n \sim 2n\log_e n$.

This analysis provides the basis of a number of insights into the behavior of the Quicksort process. Although we are more concerned with the pointers it provides about basic theoretical structures, it is worth noting quickly that our analysis already suggests when Quicksort will perform poorly. The random divisions are efficient when they almost cut the list in half, and they axe inefficient when the division is highly lopsided. It is easy to check that if one begins with a sorted list, the Quicksort process we have described would make (*n* -1) + (*n* -2) + ... + 1 = *n*(*n* -1)/2 comparisons, which is poor indeed in comparison with the expected performance on a random list.

1.3 HOW TO USE THIS SURVEY

Chapters 2 through 11 stake out a very large part of the territory at the interface of probability and algorithms. Each of these chapters is more technical than this introduction. Still, each is designed to be accessible to individuals who are considering new directions of research.

Since each of these chapters has its own introduction and abstract, browsing is encouraged. The chapters can be read independently, and overlap has been kept to a minimum.

REFERENCES

Dyer, M.E., A.M. Frieze, and C.J.H. McDiarmid (1986), On linear programs with random costs, *Math. Programming* **35**, 3-16.

Fiat, A., R. Karp, M. Luby, L. McGeoch, D. Sleator, and N. Young (1991), On competitive algorithms for paging problems, *J. Algorithms* **12**,685-699.

Karp, R.M. (1987), An upper bound on the expected cost of an optimal assignment, in *Discrete Algorithms and Complexity: Proceedings of the Japan-U.S. Joint Seminar*, D. Johnson et al., eds., Academic Press, New York.

Karp, R.M. (1991), An introduction to randomized algorithms, *Discrete Appl. Math.* **34**, 165-201.

Knuth, D.E. (1973), *The Art of Computer Programming*, vol. 3, Addison-Wesley, Reading, Mass.

MéZard, M., And G. Parisi (1987), On the solution of the random link matching problem, *J. Physique* **48**, 1451-1459.

Raghavan, P. (1990), Lecture notes on randomized algorithms, Research Report, IBM, Yorktown Heights, N.Y., unpublished.

Schwartz, J.T. (1980), Fast probabilistic algorithms for verification of polynomial identities, *J. Assoc. Comput. Mach.* **27**, 701-717.

Sedgewick, R. (1980), *Quicksort*, Garland, New York.

Seidel, R. (1992), Backwards analysis of randomized geometric algorithms, in *New Trends in Discrete and Computational Geometry*, J. Pach, ed., Springer-Verlag, New York, to appear.

Steele, J.M. (1990), Probability and statistics in the service of computer science: Illustrations using the assignment problem, *Commun. Stat*. **19**, 4315-4329.

2. Simulated Annealing

Dimitris Bertsimas and John Tsitsiklis
Massachusetts Institute of Technology

ABSTRACT

Simulated annealing is a probabilistic method proposed in Kirkpatrick et al. (1983) and Cerny (1985) for finding the global minimum of a cost function that may possess several local minima. It works by emulating the physical process whereby a solid is slowly cooled so that when eventually its structure is "frozen," it happens at a minimum energy configuration.

We restrict ourselves to the case of a cost function defined on a finite set. Extensions of simulated annealing to the case of functions defined on continuous sets have also been introduced in the literature (e.g., Gidas, 1985a; Geman and Hwang, 1986; Kushner, 1985; Holley et al., 1989; Jeng and Woods, 1990). Our goal in this review is to describe the method, its convergence, and its behavior in applications.

2.1 THE METHOD

The basic elements of **simulated annealing (SA)** are the following:

1. A finite set *S*.
2. A real-valued cost function *J* defined on *S*. Let $S^* \subset S$ be the set of global minima of the function *J*, assumed to be a proper subset of *S*.
3. For each $i \in S$, a set $S(i) \subset S - \{i\}$, called the set of neighbors of *i*.

4. For every i, a collection of positive coefficients q_{ij}, $j \in S(i)$, such that $\sum_{j \in S(i)} q_{ij} = 1$. It is assumed that $j \in S(i)$ if and only if $i \in S(j)$.
5. A nonincreasing function $T : \mathbf{N} \mapsto (0, \infty)$, called the **cooling schedule**. Here **N** is the set of positive integers, and $T(t)$ is called the **temperature** at time t.
6. An initial "state" $x(0) \in S$.

Given the above elements, the simulated annealing algorithm consists of a discrete-time inhomogeneous Markov chain $x(t)$, whose evolution we now describe. If the current state $x(t)$ is equal to i, choose a neighbor j of i at random; the probability that any particular $j \in S(i)$ is selected is equal to q_{ij}. Once j is chosen, the next state x $(t+1)$ is determined as follows:

If $J(j) \leq J(i)$, then $x(t+1) = j$.
If $J(j) > J(i)$, then
$x(t+1) = j$ with probability $\exp[-(J(j) - J(i))/T(t)]$
$x(t+1) = i$ otherwise.

Formally,

$$P[x(t+1)=j \mid x(t)=i] = q_{ij} \exp\left[-\frac{1}{T(t)} \max\{0, J(j)-J(i)\}\right] \text{ if } j \neq i,\ j \in S(i). \tag{2.1}$$

If $j \neq i$ and $j \notin S(i)$, then $P[x(t+1) = j \mid x(t) = i] = 0$.

The rationale behind the SA algorithm is best understood by considering a homogeneous Markov chain x_T (t) in which the temperature $T(t)$ is held at a constant value T. Let us assume that the Markov chain x_T (t) is irreducible and aperiodic and that $q_{ij} = q_{ji}$ for all i, j. Then $x_T(t)$ is a reversible Markov chain, and its invariant probability distribution is given by

$$\pi_T(i) = \frac{1}{Z_T} \exp\left[-\frac{J(i)}{T}\right], \quad i \in S, \tag{2.2}$$

where Z_T is a normalizing constant. (This is easily shown by verifying that the detailed balance equations hold.) It is then evident that as $T \downarrow 0$, the probability distribution π_T is concentrated on the set S^* of global minima of J. This latter property remains valid if the condition $q_{ij} = q_{ji}$ is relaxed (Faigle and Kern, 1989).

The probability distribution (2.2), known as the Gibbs distribution, plays an important role in statistical mechanics. In fact, statistical physicists have been interested in generating a sample element of *S*, drawn according to the probability distribution π_T. This is accomplished by simulating the Markov chain $x_T(t)$ until it reaches equilibrium, and this method is known as the Metropolis algorithm (Metropolis et al., 1953). In the optimization context, we can generate an optimal element of *S* with high probability if we produce a random sample according to the distribution π_T, with *T* very small. One difficulty with this approach is that when *T* is very small, the time it takes for the Markov chain $x_T(t)$ to reach equilibrium can be excessive. The SA algorithm tries to remedy this drawback by using a slowly decreasing "cooling schedule" $T(t)$.

The SA algorithm can also be viewed as a local search algorithm in which (unlike the usual deterministic local search algorithms) there are occasional "upward" moves that lead to a cost increase. One hopes that such upward moves will help escape from local minima.

2.2 CONVERGENCE ANALYSIS

2.2.1 Basic Results

Having defined the algorithm, we now address its performance. The main questions are:

1. Does $x(t)$ "converge" to the optimal set S^*?
2. How fast does the convergence to S^* take place?

The first question has been more or less completely answered, and we will now survey the main results. Much less is known about the second, as we will discuss later.

From now on, we assume that for some fixed *T* (and therefore for all *T*) the Markov chain $x_T(t)$ is irreducible and aperiodic. We say that the SA algorithm converges if $\lim_{t\to\infty} P[x(t) \in S^*] = 1$. (Note that this is convergence in probability, rather than almost sure convergence.) A fair amount of work has been concerned with finding necessary and sufficient conditions for convergence, in the above sense. The main result, due to Hajek, is presented next, following some definitions.

Theorem 2.1 (Hajek, 1988) *We say that state i communicates with* S^* at height h if there exists a path in S (with each element of the path being a *neighbor of the preceding element) that starts at i and ends at some element of S**, and such that the largest value of J along the path is J(*i*) + *h. Let d** be the smallest number such that every $i \in S$ *communicates with S** at height d*. Then, the SA algorithm converges if and only if $\lim_{t\to\infty} T(t) = 0$ *and*

$$\sum_{t=1}^{\infty} \exp[-d^*/T(t)] = \infty . \qquad (2.3)$$

The most popular cooling schedules (in theory, at least) are of the form

$$T(t) = \frac{d}{\log t}, \qquad (2.4)$$

where *d* is some positive constant. Theorem 2.1 states that SA converges if and only if $d \geq d^*$.

The constant *d** is a measure of the difficulty for *x*(*t*) to escape from a local minimum and go from a nonoptimal state to *S**. We axe primarily interested in problems where *d** > 0, which will be assumed from now on. Such problems have local minima that are not optimal. Some understanding of theorem 2.1 is provided by the following argument. Consider a local minimum whose "depth" is *d**. The SA makes an infinite number of trials to escape from it, and the probability of success at each trial is of the order of exp[-*d**/*T*(*t*)]. Then condition (2.3) amounts (by the Borel-Cantelli lemma) to saying that an infinite number of these trials will be successful. Indeed, the proof in Hajek (1988) proceeds by estimating the statistics of the exit times from certain neighborhoods of local minima.

Let π(*i; t*) = P[*x*(*t*) = *i*]. If *T*(*t*) decreases very slowly, as is the case in (2.4), then *x*(*t*) behaves, over fairly long time intervals, like a homogeneous Markov chain, and it can be reasonably expected that the difference between $\pi_{T(t)}(i)$ and π(*i; t*) is small at all times. Indeed, one of the very first convergence proofs (Geman and Geman, 1984) was based on this idea, although the results therein were less sharp than theorem 2.1.

In order to acquire more intuition about the interpretation of theorem 2.1, we will carry the connection between SA and the corresponding family of homogeneous Markov chains further. For this goal we consider the cooling schedule *T*(*t*) = *d*/log *t*. In general, the statistics of the Markov chain *x*(*t*) under a slowly varying cooling schedule *T*(*t*) remain pretty much unchanged if a related cooling schedule is used in which the temperature is held constant for fairly long time periods. In our case, the schedule *T*(*t*) = *d*/log *t* can be approximated as follows. Let $t_1 = 1$ and $t_{k+1} = t_k + \exp(kd)$. Then let $|\hat{T}(t) = 1/k|$, for $t_k \leq t < t_{k+1}$. Consider the *k*th segment [t_k, t_{k+1}] of the piecewise constant schedule $\hat{T}(t)$. We are dealing with the reversible homogeneous chain *x*1/ *k*(*t*), and we will now investigate how fast it reaches steady state.

*We want to study the convergence of the chain x*1/*k*(*t*). The eigenvalues of its transition-probability matrix are real. Its relaxation time is determined by its second-largest eigenvalue λ_2 for which good estimates are available, at least in the limit as $k \to \infty$ (see, e.g., Ventcel, 1972; Chiang and Chow, 1988; and Holley and Stroock, 1988). In particular, if the cost function *J* has a unique global minimum, the relaxation time $|\log(1 - \lambda_2)|$ is well approximated by exp(*kd**). Interestingly enough, this is the same as constant *d** defined in theorem 2.1, something that is far from obvious. This yields another interpretation of the convergence condition $d > d^*$ for the schedule $\hat{T}(t)$. If $d < d^*$, then at each temperature 1/*k* we are running *x*1/*k*(*t*) for a negligible fraction of its relaxation time, and this is not enough for π(*i; t*) to stay close to $\pi_T(i)$. On the other hand, if *d* > *d**, then the interval [t_k, t_{k+1}] corresponds to exp[*k*(*d**-*d*)] relaxation times of *x*1/*k*(*t*), which implies that π(*i;* t_{k+1}) is very close to π1/*k*(*i*), as $k \to \infty$.

One can also pursue this approximation by piecewise constant schedules directly, without introducing eigenvalue estimates. The main idea is that, at low temperatures, the statistics of a homogeneous chain can be accurately bounded by viewing it as a singularly perturbed Markov chain and using rather crude large-deviation bounds (Tsitsiklis, 1988, 1989). For other convergence proofs, see Connors and Kumar (1988), Gidas (1985b), Mitra et al. (1986), and Honey and Stroock (1988).

The convergence of SA (in the sense defined earlier in this section) is a reassuring property, but it is fax from enough for SA to be a useful algorithm. We also need to know the speed of convergence. It can be shown that for any schedule *T*(*t*) = *d/*log *t*, and for all *t*,

$$\max_{x(0)} \mathrm{P}[x(t) \notin S^* \mid x(0)] \geq A/t^a , \qquad (2.5)$$

where *A* and *a* are positive constants depending on the function *J* and the neighborhood structure. If we wish *x*(*t*) to be outside *S** with probability less than ϵ, we need $t \geq (A/\epsilon)^{1/a}$.

We now turn our attention to the practical relevance of some of the convergence results. From a more practical perspective, while the algorithm is being run, one should keep track of the best state *i* encountered so fax and its associated cost *J*(*i*). If the algorithm is to be run for *t** time steps, we are not really interested in the value of $\mathrm{P}(x(t^*) \notin S^*)$. Rather, we are interested in the probability that no state in S^* is visited during the execution of the algorithm. Given a cooling schedule of the form $T(t) = d/\log t$, with $d > d^*$, it can be shown that this probability is at most $A/(t^*)^a$, for some positive constants A and a. It vanishes as $t \to \infty$, which might seem encouraging. On the other hand, if the temperature is fixed at any positive value (or even at infinity, corresponding to a random walk), the probability that no state in S^* is visited in t^* time units is at most $Be^{-t^*/b}$, for suitable positive constants B and b. So, for very large times, the performance of a random walk would seem to be better than the performance guarantees of simulated annealing. The key point is that the above analyses, based on time asymptotics, involve extremely large constants and are largely irrelevant. Indeed the constants in the above estimates are often much larger than the cardinality of the state space S. In particular, the above analysis cannot even establish that simulated annealing is preferable to exhaustive search.

2.2.2 Taking the Instance Size into Account

One can trace the inadequacy of the analytical methods mentioned so far to the fact that they deal with one instance at a time. A more realistic approach would be to consider a family of instances, together with a notion of instance size, and then study the statistics of the first hitting time of the set S^* as a function of the instance size. (We refer to such results as **complexity results**.) This would then provide a meaningful yardstick for comparing simulated annealing to alternative methods. As an extension, if we are only interested in approximately optimal solutions to a given problem, we can define a set S^* of approximately optimal solutions and study the statistics of the first hitting time of S^*. Unfortunately, such results have proved very difficult to derive. We review briefly some of the progress that has been made along this direction.

All available positive complexity results we are aware of use a schedule $T(t)$ in which the temperature is held constant in time, although it may depend on the instance size. The only nontrivial combinatorial problem for which SA has been rigorously analyzed is the maximum matching problem. It has been shown by Sasaki and Hajek (1988) that, for bounded-degree graphs with n nodes, the expected time until SA hits an optimal configuration is $O(n^5)$. This is slower than other available algorithms, but the result is encouraging. On the other hand, no algorithm of the simulated annealing type (even with time-varying temperature) can solve the matching problem in polynomial expected time when the degree bound is relaxed (see Sasaki and Hajek, 1988).

Hajek (1985) studied an example in which the state space is $S = \{1, 2, 3\}^N$ and the cost of a typical state $s = (s_1, \dots , s_N)$ is equal to $\sum_{i=1}^{N} V(s_i)$. This corresponds to a minimization in N identical uncoupled problems, and the optimal solution is evident. It is argued that if the temperature is suitably chosen as a function of N, the expected first hitting time of the global minimum grows polynomially with N. This is better than exhaustive search, because the cardinality of S is 3^N. It is also better than the "local search with multistart" method, whereby a state in S is chosen repeatedly at random and each time a local search (pure descent) algorithm is run until a local minimum is reached (Hajek, 1985). Of course, in this example a much more efficient algorithm exists. What is encouraging is that SA does not really use the information that the cost is separable, i.e., that there is no coupling between s_i and s_j. For this reason, one may hope that simulated annealing would run in polynomial time even if the problem was perturbed by introducing some sufficiently weak coupling between components. No such extensions are currently available.

One reason why SA works well in the preceding example is that if a given state is far from optimal, then there exists a large number of paths that lead to another state with lower cost. The probability that the state $x(t)$ escapes from a local minimum of depth d along any particular path, and in a single trial, is at most $\exp(-d/T)$. On the other hand, if the number of candidate paths is very large, the probability of escape is substantial. It is here that the combinatorial structure of the state space S and its neighborhood system should become significant.

In interesting combinatorial settings the state space is usually exponential in the instance size. So constant-temperature SA would work in polynomial time if the relaxation time were polylogarithmic in the size of the state space ("rapid mixing"). There are a number of available positive results on rapid mixing in Markov chains, but they deal mostly with the case where $T = \infty$, corresponding to a random walk. (This is a much simpler special case, because the cost function J has no effect.) Unfortunately, simulated annealing becomes interesting at the opposite end, when T is very small. Proving rapid mixing for large SA Markov chains at small temperatures is a challenging task.

One class of problems for which there is some hope of obtaining positive complexity results arises in the context of image processing. Here, we have an $N \times N$ grid. To each gridpoint (i, j), we associate a variable s_{ij} taking values in a finite set *A*. We thus obtain a configuration space $S = A^{N^2}$. Many image processing and pattern recognition problems lead to a cost function $J : S \mapsto \mathbf{R}$ of the form

$$J(s) = \sum_{(i,j)} f_{ij}(s_{ij}) + \sum_{(i,j),\ (k,\ell)} g_{ij,k\ell}(s_{ij}, s_{k\ell}),$$

where $g_{ij,k\ell}$ is identically zero unless (*i, j*) and (k, ℓ) are neighboring grid-points. Starting with Geman and Geman (1984), simulated annealing has become a very popular method for such problems. Here one defines two states (configurations) to be neighbors if they differ only at a single grid-point. Note that when a configuration change is contemplated (that is, a change of some s_{ij}), the cost difference (which determines the probability of accepting the change) depends only on the gridpoints neighboring (*i, j*). For this reason, the evolution of the configuration can be viewed as the time evolution of a Markov random field. The relaxation times of Markov random fields have been extensively studied (see, e.g., Liggett, 1988), but under rather special assumptions on the functions f_{ij} and $g_{ij,k\ell}$. Thus, the available results are not yet applicable to the cost functions that arise in image processing.

As far as theory is concerned, there is at present a definite lack of rigorous results justifying the use of simulated annealing. Even if SA is accepted, there are no convincing theoretical arguments favoring the use of time-varying (decreasing) cooling schedules, as opposed to the use of a constant temperature. (This latter question is partially addressed in Hajek and Sasaki (1989).)

2.3 BEHAVIOR IN PRACTICE

Despite the lack of a rigorous theoretical justification of its speed of convergence, researchers have used SA extensively in the last decade. There are numerous papers discussing applications of SA to various problems. We have already mentioned that SA is extensively used in image processing. In order to give an indication of its performance, we will review some of the work concerning the application of SA to combinatorial optimization problems.

In a comprehensive study of SA, Johnson et al. (1990, 1991, 1992) discuss the performance of SA on four problems: the traveling salesman problem (TSP), graph partitioning problem (GPP), graph coloring problem (GCP), and number partitioning problem (NPP). Johnson et al. apply SA to these NP-hard problems using a cooling schedule in which the temperature decreases geometrically; i.e., $T(t + 1) = rT(t)$. In general, the performance of SA was mixed: in some problems, it outperformed the best known heuristics for these problems, and, in other cases, specialized heuristics performed better. More specifically:

1. In the graph partitioning problem, a graph $G = (V, E)$ is given, and we are asked to partition the set of vertices V into two subsets V_1 and V_2 so as to minimize

   $$|X| + \lambda(|V_1| - |V_2|)^2,$$

   where X is the set of edges joining a node in V_1 with a node in V_2 , and λ is a weighting factor. For the GPP, SA obtains final solutions that are at best some 5 percent better than those obtained by the best of the more traditional algorithms (e.g., the Kernighan and Lin (1970) heuristic) if the latter are allowed the same amount of computation time as SA. For sparse graphs, SA was better than repeated applications of the Kernighan-Lin heuristic, which is based on ideas of local optimization, whereas for some structured graphs the Kernighan-Lin heuristic was better.
2. In the graph coloring problem, a graph $G = (V, E)$ is given, and we are asked to partition the set of vertices into a minimal number of subsets, such that no edge has both endpoints in the same subset. For the graph coloring problem, SA produces final solutions that are competitive with those obtained by a tailored heuristic (the one by Johri and Matula (1982)), which is considered the best one for this problem. However, computation times for SA are considerably longer than those of the specialized heuristic.
3. For the traveling salesman problem, SA consistently outperforms solutions found by repeated application of iterative improvement, based on 2-opt or 3-opt transitions, but it is a consistent loser when compared with the well-known algorithm of Lin and Kernighan (1973). The latter is based on k-opt transitions, and at each iteration it decides dynamically the value of k.

Another interesting point is that the choice of the cooling schedule influences the quality of solution obtained. In Laarhoven and Aarts (1987) the authors compare three different cooling schedules for the graph partitioning problem, and they observe that the quality of the solution found by the different cooling schedules can differ as much as 10 percent. Another observation is that the computation times can be excessive for some problems.

In addition to the above mentioned developments in image processing, SA and various alternative versions based roughly on it have been used in statistical applications. Bohachevsky et al. (1986) proposed a "generalized" SA procedure for continuous optimization problems and applied their method to an optimal design problem. Many researchers have considered SA as a tool in the development of optimal experimental designs. Recent examples include Currin et al. (1991), Meyer and Nachtsheim (1988), and Sacks and Schiller (1988). Variants of SA based on Bayesian ideas have been proposed by Laud et al. (1989) and van Laarhoven et al. (1989).

Overall, SA is a generally applicable and easy-to-implement probabilistic approximation algorithm that is able to produce good solutions for an optimization problem, even if we do not understand the structure of the problem well. We believe, however, that more research, both theoretical and experimental, is needed to assess further the potential of the method.

REFERENCES

Bohachevsky, I., M.E. Johnson, and M.L. Stein (1986), Generalized simulated annealing for function optimization, *Technometrics* **28**, 209-217.

Cerny, V. (1985), A thermodynamic approach to the traveling salesman problem: An efficient simulation, *J. Optim. Theory Appl.* **45**, 41-51.

Chiang, T.-S., and Y. Chow (1988), On eigenvalues and optimal annealing rate, *Math. Oper. Res.* **13**, 508-511.

Connors, D.P., and P.R. Kumar (1988), Balance of recurrence order in time-inhomogeneous Markov chains with applications to simulated annealing, *Prob. Eng. Inform. Sci.* **2**, 157-184.

Currin, C., T. Mitchell, M. Morris, and D. Ylvisaker (1991), Bayesian prediction of deterministic functions, with applications to the design and analysis of computer experiments, *J. Amer. Star. Assoc.* **86**, 953-963.

Faigle, U., and W. Kern (1989), Note on the Convergence of Simulated Annealing Algorithms, Memorandum 774, University of Twente, Enschede, Netherlands.

Geman, S., and D. Geman (1984), Stochastic relaxation, Gibbs' distribution and the Bayesian restoration of images, *IEEE Trans. Pattern Anal. Mach. Intell*. **6**, 721-741.

Geman, S., and C.-R. Hwang (1986), Diffusions for global optimization, *SIAM J. Contr. Optim*. **24**, 1031-1043.

Gidas, B. (1985a), Global optimization via the Langevin equation, in *Proc. 24th IEEE Conference on Decision and Control*, IEEE, New York, 774-778.

Gidas, B. (1985b), Nonstationary Markov chains and convergence of the annealing algorithm, *J. Stat. Phys*. **39**, 73-131.

Hajek, B. (1985), A tutorial survey of theory and applications of simulated annealing, in *Proc. 24th IEEE Conference on Decision and Control*, IEEE, New York, 755-760.

Hajek, B. (1988), Cooling schedules for optimal annealing, *Math. Oper. Res*. **13**, 311-329.

Hajek, B., and G. Sasaki (1989), Simulated annealing—to cool or not, *Syst. Contr. Lett*. **12**, 443-447.

Holley, R.A., and D.W. Stroock (1988), Simulated annealing via Sobolev inequalities, *Commun. Math. Phys*. **115**, 553-569.

Holley, R.A., S. Kusuoka, and D.W. Stroock (1989), Asymptotics of the spectral gap with applications to the theory of simulated annealing, *J. Funct. Anal*. **83**, 333-347.

Jeng, F.-C., and J.W. Woods (1990), Simulated annealing in compound Gaussian random fields, *IEEE Trans. Inform. Theory* **36**, 94-107.

Johnson, D., C. Aragon, L. McGeoch, and C. Schevon (1990), Optimization by simulated annealing: An experimental evaluation, Part I: Graph partitioning, *Oper. Res*. **37**, 865-892.

Johnson, D., C. Aragon, L. McGeoch, and C. Schevon (1991), Optimization by simulated annealing: An experimental evaluation, Part II: Graph coloring and number partitioning , *Oper. Res*. **39**, 378-406.

Johnson, D., C. Aragon, L. McGeoch, and C. Schevon (1992), Optimization by simulated annealing: An experimental evaluation, Part III: The traveling salesman problem, in preparation.

Johri, A., and D. Matula (1982), Probabilistic bounds and heuristic algorithms for coloring large random graphs, Technical Report 82-CSE-6, Department of Computer Science and Engineering, Southern Methodist University, Dallas, Tex.

Kernighan, B., and S. Lin (1970), An efficient heuristic procedure for partioning graphs, *Bell Syst. Tech. J*. **49**, 291-307.

Kirkpatrick, S., C.D. Gelett, and M.P. Vecchi (1983), Optimization by simulated annealing, *Science* **220**, 621-630.

Kushner, H.J. (1985), Asymptotic global behavior for stochastic approximations and diffusions with slowly decreasing noise effects: Global minimization via Monte Carlo, Technical Report LCDS 85-7, Brown University, Providence, R.I.

Laud, P., L.M. Berliner, and P.K. Goel (1989), A stochastic probing algorithm for global optimization, in *Proc. 21st Symposium on the Interface between Computer Science and Statistics*. Revised version to appear in *J. Stoch. Opt*. (1992).

Liggett, T. (1988), *Interacting Particle Systems*, Springer-Verlag, New York.

Lin, S., and B. Kernighan (1973), An effective heuristic algorithm for the traveling salesman problem, *Oper. Res*. **21**, 498-516.

Metropolis, N., A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller (1953), Equation of state calculations by fast computing machines, *J. Chem. Phys*. **21**, 1087-1092.

Meyer, R.K., and C.J. Nachtsheim (1988), Constructing exact D-optimal experimental designs by simulated annealing, *Amer. J. Math. Manage. Sci*. **8**, 329-359.

Mitra, D., F. Romeo, and A. Sangiovanni-Vincentellim (1986), Convergence and finite-time behaviour of simulated annealing, *Adv. Appl. Probab* . **18**, 747-771.

Sacks, J., and S. Schiller (1988), Spatial designs, in *Fourth Purdue Symposium on Statistical Decision Theory and Related Topics* , S.S. Gupta, ed., Academic Press, New York.

Sasaki, G., and B. Hajek (1988), The time complexity of maximum matching by simulated annealing, *J. Assoc. Comput. Mach.* **35**, 387-403.

Tsitsiklis, J.N. (1988), A survey of large time asymptotics of simulated annealing algorithms, in *Stochastic Differential Systems, Stochastic Control Theory and Applications*, W. Fleming and P.L. Lions, eds., Springer-Verlag, 583-599.

Tsitsiklis, J.N. (1989), Markov chains with rare transitions and simulated annealing, *Math. Oper. Res*. **14**, 70-90.

van Laarhoven, P., and E. Aarts (1987), *Simulated Annealing: Theory and Applications*, D. Reidel, Dordrecht, Netherlands.

van Laarhoven, P., C. Boender, E. Aarts, and A. Rinnooy Kan (1989), A Bayesian approach to simulated annealing, *Probab. Eng. Inform. Sci.* **3**, 453-475.

Ventcel, A.D. (1972), On the asymptotics of eigenvalues of matrices with elements of order $\exp\{-V_{ij}/(2\epsilon^2)\}$, *Dokl. Akad. Nauk SSSR* **202**, 65-68.

3. Approximate Counting Via Markov Chains

David Aldous
University of California at Berkeley

ABSTRACT

For large finite sets where there is no explicit formula for the size, one can often devise a randomized algorithm that approximately counts the size by simulating Markov chains on the set and on recursively defined subsets.

3.1 EXACT AND APPROXIMATE COUNTING

For a finite set *S*, there is a close connection between

1. having an explicit formula for the size |*S*| and
2. having a bounded-time algorithm for generating a uniform random element of *S*.

As an elementary illustration, we all know that there are *n*! permutations of *n* objects. From a proof of this fact, we could write down an explicit 1-1 mapping *f* between the set of permutations and the set $A = \{(a_1 , a_2, \ldots ,$ $a_n)$: $A=\{(a_1,a_2,\ldots,a_n):1\leq a_i\leq i\}$. Then we could simulate a uniform random permutation by first simulating a uniform random element *a* of *A* and then computing *f*(*a*). Conversely, given an algorithm that was guaranteed to produce a uniform random permutation after *k*(*n*) calls to a random number generator, we could (in principle) analyze the working of the algorithm in order to calculate the chance p of getting the identity permutation. Then we can say that the number of permutations equals $1/p$.

But now suppose we have a hard counting problem for which we cannot find an explicit formula for $|S|$. The purpose of this chapter is to describe a remarkable technique for constructing randomized algorithms that count S *approximately*. This technique, developed in recent years, has two ingredients. The first idea is that in some settings, having an algorithm for generating an *approximately* uniform random element of S can be used recursively to estimate *approximately* the size $|S|$. We illustrate this with two examples in the next section. The second idea is that we can obtain an approximately uniform random element of S by running a suitable Markov chain on state-space S for sufficiently many steps. This idea and the particular chains used in the two examples are discussed in §3.3. The latter idea is fundamentally similar to the use of Markov chains in the Metropolis algorithm, which underlies the simulated annealing algorithm discussed in Chapter 2. The difference is that here we seek to simulate a *uniform* distribution rather than a distribution that is deliberately biased towards minima of a cost function. Our Markov chains are simpler to analyze and permit rigorous discussion of polynomial-time convergence issues that seem too hard for rigorous treatment in the context of simulated annealing.

3.2 RECURSIVE ESTIMATION OF SIZE

Example: Volume of a Convex Set (Dyer et ah, 1989, 1991). Consider the problem of estimating the volume vol(K) of a convex set K in n-dimensional space, for large n. Suppose we are told that $B(1) \subset K \subset B(r)$, where $B(r)$ is the ball of radius $r = r(n)$. It is believed that there is no polynomial-time deterministic algorithm to approximate vol(K). But suppose we have a means of simulating, at unit cost, a random point in K whose distribution is approximately uniform. Specify some subset K_1 with $B(1) \subset K_1 \subset K$ and for which vol(K_1)/vol(K) is not near 0 or 1. Then by simulating m random points in K and counting the proportion $p(1)$ that fall within K_1,

$$p(1) = \text{vol}(K_1)/\text{vol}(K) + \text{ bias } + \text{ sampling error}\,.$$

Now specify a decreasing sequence of such convex subsets

$$K = K_0 \supset K_1 \supset K_2 \ldots \supset K_L = B(1)$$

that will have length $L = O(n \log r)$. Then as above we can estimate each ratio $\mathrm{vol}(K_i)/\mathrm{vol}(K_{i-1})$ as $p(i)$, and finally

$$\mathrm{vol}(B(1)) \times \prod_{i=1}^{L} \frac{1}{p(i)} \text{ is an approximation to } \mathrm{vol}(K). \qquad (3.1)$$

How accurate is this approximation? The elementary mathematics of random sampling shows that the sampling error at each step is $O(m^{-1/2})$. So by choosing m to be a large multiple of L^2, the sampling error in (3.1) is made small. Note this makes the total cost $O(L^3)$. The remaining issue is to find a way of simulating an approximately uniform point in K with bias $o(1/L)$, where *bias* is formalized in (3.2). We return to this issue in the next section.

Example: Matchings in a Graph (Sinclair and Jerrum, 1989). Fix a graph G with n vertices. In graph-theory terminology,two edges are **independent** if there is no common vertex, and a **matching** is a set of independent edges. Let $M(G)$ be the set of all matchings of G. How can we estimate $|M(G)|$, the number of matchings? Let us show how to do so, given a method of simulating approximately uniform random matchings.

Enumerate the vertices as $s_1, s_2, \ldots, s_n$. Let G_i be the subgraph obtained by deleting $s_1, \ldots, s_i$. If we could simulate a random matching $\mathcal{M}$ that was exactly uniform (i.e., equally likely to be each of the $|M(G)|$ matchings), then

$$P(s_1 \text{ is in no edge of } \mathcal{M}) = |M(G_1)|/|M(G)|.$$

So given a method of simulating an approximately uniform matching, we can do m such simulations and record the proportion $p(1)$ of these m simulations in which s_1 is not an edge of the random matching. Then

$$p(1) = |M(G_1)|/|M(G)| + \text{ bias } + \text{ sampling error}.$$

Similarly, do m simulations of an approximately uniform random matching in G_{i-1}, and use the proportion $p(i)$ in which s_i is not an edge of the matching as an estimator of $|M(G_i)|/|M(G_{i-1})|$. This argument, analogous to (3.1), leads to

$$\prod_i 1/p(i) \text{ is an approximation to } |M(G)|.$$

These examples share a **self-reducibility** property formalized in Sinclair and Jerrum (1989). Informally, we can relate the size of the set S under consideration to the size of a smaller set S_1 of the same type. The examples illustrate the first idea stated in §3.1. That is, with self-reducible sets an algorithm for generating an *approximately* uniform random element can be used recursively to estimate *approximately* the size $|S|$.

Three points deserve mention:

1. In the two examples, the self-reducibility property was exact. The reader will rightly suspect that such examples are rare. But the method has been successfully applied where there is only some cruder notion of reducibility, for instance to the problems of approximating the permanent (Jerrum and Sinclair, 1989) and of counting regular graphs (Jerrum and Sinclair, 1988).
2. We should emphasize the point of the product-form estimators: they enable us to estimate exponentially small probabilities in polynomial time. Thus in the first example both vol(K)/vol(B(r)) and vol(B(1))/ vol(K) are typically $O(r\text{-}n)$, so to estimate the ratio in one step would require simulating $O(r^n)$ points instead of $O(n\log r)^3$ points.
3. In principle, we could combine this self-reducibility technique with *any* method of simulating approximately uniform elements of the set, not just the Markov chain method in the next section. But in practice no other methods are known for the examples where the technique has been successful.

3.3 THE MARKOV CHAIN METHOD OF SIMULATING A PROBABILITY DISTRIBUTION

This method is a specialization of the Metropolis algorithm described in Chapter 2 on simulated annealing. Suppose we want to simulate the uniform probability distribution π on a large finite set S. Suppose we cau define a Markov chain (picture a randomly moving particle) on states S whose stationary distribution is π. After running the chain for a sufficiently large number of steps, the position of the particle will be approximately uniform. The simplest way to define such a chain is to define a regular graph structure with vertex-set S. In this graph setting, let the Markov chain be a simple random walk on the graph, at each step moving from the current vertex s to a vertex s' chosen uniformly from the neighbors of s.

We now describe graphs for our two examples. To fit the first example into this framework, we discretize in the obvious way, replacing the convex set *K* by its intersection $S = K \cap \delta Z^n$ with the lattice of edge-length δ, for some small δ. By simulating random walk on *S* for sufficiently many steps, we will get to a point approximately uniform in *S*, and hence approximately uniform in *K*.

In the second example, construct a Markov chain as follows. From a matching $\mathcal{M}_0$, move to a new matching $\mathcal{M}_1$ as follows:

1. Pick an edge (*v, w*) of *G* at random.
2. If (*v, w*) is an edge of $\mathcal{M}_0$, delete it.
3. If *v* and *w* are singletons of $\mathcal{M}_0$, add the edge (*v, w*).
4. Otherwise, *v* (say) is a singleton and *w* is in an edge (*w, u*) of $\mathcal{M}_0$: delete the edge (*w, u*) and add the edge (*v, w*).

After sufficiently many steps, the random matching will have approximately uniform distribution.

3.4 ESTIMATING MIXING TIMES

There is a glaring gap to bridge before we can use the Markov chain method as an honest randomized algorithm. In the simplest use of this method, we will run the chain for some number *k* of steps and employ the final state as our "approximately uniform" random element. But how many steps are enough? To justify algorithms, we need bounds on a **mixing time** such as τ_1 or τ_2 below, which formalize the intuitive idea of "number of steps until the distribution is approximately the stationary distribution, independent of starting position." The classical elementary theory of Markov chains says that, under natural conditions (irreducible, aperiodic), the distribution of *k* steps converges as $k \to \infty$ to the stationary distribution π. But proofs of this elementary general result do not lead to useful bounds on in particular examples. The invention of approximate counting, as well as other Markov chain applications such as card-shuffling (Bayer and Diaconis, 1992; Diaconis, 1988) and queueing theory (Blanc, 1988), have motivated recent work of the mathematical issue of estimating mixing times.

In the context of approximate counting, we have some flexibility in the exact choice of the Markov chain, and it is usually possible and very convenient to choose the chain to be **reversible**, that is, to satisfy $\pi(i)P(i, j) =$

$\pi(j)P(j,\ i)$ for all states *i, j*. A classical way of formalizing rates of convergence is via the second-largest eigenvalue $\lambda < 1$ of the transition matrix *P*. This leads to one definition of mixing time as

$$\tau_2 = 1/(1-\lambda)\,.$$

Another definition of mixing time, more directly applicable in our setting, uses the time taken to make the maximal bias small:

$$\tau_1 = \min\{n : |P_i(X_n \in A) - \pi(A)| \le 1/e \text{ for all } i \in S, A \subset S\}\,. \qquad (3.2)$$

It is easy to show that after $m\tau_1$ steps the bias is at most 2*e-m*, so that a bound on τ_1 enables us to specify confidently a number of steps guaranteed to reduce the bias sufficiently in our applications to approximate counting.

There are several other possible definitions of mixing times, and easy general inequalities giving relations between them. The real issue is not the choice of definition but the development of widely applicable techniques that enable *some* mixing time to be bounded. One interesting technique is to relate mixing times to the quantity

$$c(P) \equiv \max_{A \subset S} \frac{\pi(A)(1-\pi(A))}{\sum_{i \in A, j \notin A} \pi(i)P(i,j)}\,.$$

From a bound on *c*(*P*), one can use **Cheeger's inequality** (e.g., Sinclair and Jerrum, 1989) to bound τ_2 and then obtain the desired bound on τ_1. Such results are theoretically appealing in that *c*(*P*) relates to the "geometry" of the chain, and (for simple random walk on a graph *G*) to isoperimetric inequalities on *G*, a quantity of intrinsic graph-theoretic interest. On the other hand, it is usually hard to estimate *c*(*P*) well. Diaconis and Stroock (1991) argue that to get upper bounds on τ_2 one can do as well by a direct use of a **distinguished paths** method. For each pair of states, one specifies a path connecting them, seeking to minimize the number of times any fixed edge is used: one can then bound τ_2 in terms of the maximal "probability flow" along any edge. This is still an active research area, and doubtless further techniques will be developed.

We said that the simplest use of this method was to run the chain for some prespecified number of steps and employ the final state as our "approximately uniform" random element. As described in §3.2, this process is then repeated *m* times (say) in order to calculate the proportion of these *m* final values that fall in a given subset. Many variations are possible. For instance, instead of the *m* repetitions one could run the chain for the same total time but estimate the desired probability of a subset by looking at the proportion of *all* times (except an initial segment) that the chain spent in the subset. This is likely to give somewhat of an improvement in practice, though in principle may be no better (Aldous, 1987).

3.5 CONCLUSION

The topic of approximate counting is still under vigorous development. Let us close with one direction for future research. In addition to the familiar counting problems from the theory of combinatorial algorithms, there are a range of interesting problems arising from physics. We mentioned that the Markov chain method was just a specialization of the Metropolis algorithm for simulating a given probability distribution by inventing a suitable Markov chain. Physicists have long used that algorithm for simulation but typically have been unable to justify rigorously the results of the simulation by proving bounds on the mixing time. For instance, physicists have studied self-avoiding random walks rather intensely by simulation and by nonrigorous mathematical methods. A celebrated open problem is to prove a polynomial-time bound for the mixing time in some algorithm for simulating self-avoiding walks in three dimensions: equivalently, to give a polynomial-time randomized algorithm for approximately counting the number of self-avoiding walks of length *n*. Bringing together the users of randomized algorithms in physics and the "theoretical" researchers in computer science promises to be a fruitful collaboration.

REFERENCES

Aldous, D.J. (1987), On the Markov chain simulation method for uniform combinatorial distributions and simulated annealing, *Probah. Eng. Inform. Sci.* **1**, 33-46.

Bayer, D., and P. Diaconis (1992), Trailing the dovetail shuffle to its lair, *Ann. Appl. Probab*. **2**, to appear.

Blanc, J.P.C. (1988), On the relaxation times of open queueing networks, in *Queueins Theory and Its Applications*, CWI Monographs 7, North-Holland, New York, 235-259.

Diaconis, P. (1988), *Group Representations in Probability and Statistics* , Institute of Mathematical Statistics, Hayward, Calif.

Diaconis, P., and D. Stroock (1991), Geometric bounds for eigenvalues of Markov chains, *Ann. Appl. Probab*. **1**, 36-61.

Dyer, M., A. Frieze, and R.. Kannan (1989), A random polynomial time algorithm for approximating the volume of convex bodies, in *Proc. 21st ACM Symp. on Theory of Computing*, ACM Press, New York, 375-381.

Dyer, M., A. Frieze, and R. Kannan (1991), A random polynomial time algorithm for approximating the volume of convex bodies, *J. Assoc. Comput. Mach*. **38**, 1-17.

Jerrum, M., and A. Sinclair (1988), Fast uniform generation of regular graphs, *Theor. Comput. Sci*. **73**, 91-100.

Jerrum, M., and A. Sinclair (1989), Approximating the permanent, *SIAM J. Comput*. **18**, 1149-1178.

Sinclair, A., and M. Jerrum (1989), Approximate counting, uniform generation and rapidly mixing Markov chains, *Inform. Comput*. **82**, 93-133.

4. Probabilistic Algorithms for Speedup

Joan Feigenbaum and Jeffrey C. Lagarias
AT&T Bell Laboratories

ABSTRACT

This chapter surveys situations in which probabilistic algorithms offer speedup over what is possible using deterministic algorithms, either in practice or in principle.

4.1 INTRODUCTION

One of the most compelling reasons to use randomized algorithms is that they permit certain problems to be solved faster than is possible by deterministic methods.

One pays a price for such speedup, which is the possibility of occasional very long computations or of occasional errors in the computation. The amount of possible speedup may depend on whether the algorithm is required to give either a correct answer or no answer, or whether it is permitted to give incorrect answers.

Section 4.2 describes a formal version of such questions from the viewpoint of computational complexity theory, in terms of probabilistic complexity classes. The question of whether probabilistic algorithms can give a superpolynomial speedup over deterministic algorithms is restated there as the unsolved problem, does $\mathrm{P} \neq \mathrm{BPP}$? Thus it is unresolved whether or not speedup exists, and settling this question is likely to be as hard as the notorious $P \neq NP$? problem of complexity theory.

There is a weaker sense in which probabilistic algorithms currently offer advantages over deterministic ones. There exist problems for which the fastest known algorithm is probabilistic and problems for which at present the fastest fully analyzable algorithm is probabilistic. Here the situation reflects the current limitations on our knowledge. Section 4.3 describes examples of such problems from number theory, i.e., primality testing and factorization. In both cases, at present, the best fully analyzed algorithms are probabilistic ones.

Probabilistic algorithms give a demonstrable speedup in the exchange of information among several people, each having partial information to begin with. This is the subject matter of communication complexity. It plays an important role in distributed computing, in which exchange of information among the processors is an important part of the computational task. Communication complexity results are described in §4.4.

4.2 PROBABILISTIC COMPLEXITY CLASSES

The theoretical possibility of speedup has been formalized in computational complexity theory with the notion of **probabilistic complexity classes;** these classify computational problems according to the running times and types of error allowed by probabilistic algorithms. Here we define several of these complexity classes.

The model of computation used is a probabilistic Turing machine. This is a Turing machine that has an extra "random" tape containing a random string of zeros and ones that can be read as necessary. A **probabilistic polynomial-time Turing machine (PPTM)** is such a machine equipped with a clock that, when given an input of *n* bits, always halts after *p*(*n*) steps, where *p* is a fixed polynomial. The performance of such machines is averaged over the uniform distribution of all random bits read by the machine. (At most, *p*(*n*) random bits are read during the computation on an *n*-bit input.) The computational problems considered are those of recognizing a language $\mathcal{L} \subseteq \{0,1\}^*$, where {0, 1}* denotes the set of all finite strings of zeros and ones. For example, $\mathcal{L}$ might consist of all prime numbers, expressed using their binary representations.

We say that $\mathcal{L}$ is in the complexity class **BPP (bounded-error-probability polynomial time)** if there is a PPTM that, given $b \in \Omega$, outputs

“$x \in \mathcal{L}$” with probability at least 3/4, averaged over all random tapes for the fixed input *x*, and given $x \notin \mathcal{L}$ outputs “$x \notin \mathcal{L}$” with probability at least 3/4 similarly. In this complexity class, two-sided classification errors are allowed: The machine may output “$x \in \mathcal{L}$” when in fact $x \notin \mathcal{L}$, and vice versa. Note that any problem in BPP is solvable by a PPTM for which the error probability is exponentially small. The new PPTM repeats the test for $x \in \{0,1\}^n$ on the old PPTM *n* times using independent random bits each time, and then takes a majority vote on the *n* outputs to decide whether $x \in \mathcal{L}$; the error probability for the new PPTM is less than (3/4) *n*/2.

We say that $\mathcal{L}$ is in the complexity class **RP (random polynomial time)** if there is a PPTM that, when given an input $x \in \mathcal{L}$, outputs “$x \in \mathcal{L}$” at least 3/4 of the time and, when given an input $x \notin \mathcal{L}$, always outputs “$x \notin \mathcal{L}$.” . That is, only one-sided error is allowed. We say that $\mathcal{L}$ is in the complementary complexity class **co-RP (co-random polynomial time)** if there is a PPTM that, when given an input $x \notin \mathcal{L}$, outputs “$x \notin \mathcal{L}$” at least 3/4 of the time and, when given an input $x \in \mathcal{L}$, always outputs “$x \in \mathcal{L}$.”

Finally, we say $\mathcal{L}$ is in **ZPP (zero-error-probability polynomial time)** if $\mathcal{L}$ is in both RP and co-RP. In this case, when the PPTM outputs either “$x \in \mathcal{L}$” or “$x \notin \mathcal{L}$”, it is correct, but it is allowed to give no output for a small fraction of random tapes.

The complexity class BPP is closed under complementation, and we have the following inclusions shown in Figure 4.1, where P denotes the set of (deterministic) polynomial-time recognizable languages.

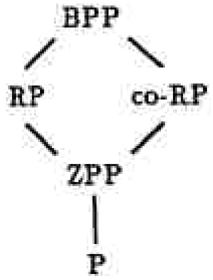

FIGURE 4.1: Relationships between complexity classes.

In this framework, it is unknown whether probabilistic algorithms ever give a superpolynomial speedup, i.e., whether there are any languages in

BPP that are not in P. Yao (1982) announced a converse result suggesting that at most a subexponential speedup is possible:

Theorem 4.1 (Yao, 1982) *If there exists a uniform secure polynomial pseudorandom bit generator, then*

$$\mathrm{BPP} \subseteq \bigcap_{\epsilon > 0} \mathrm{DTIME}\left(2^{n^{\epsilon}}\right).$$

A proof of this result appears in Boppana and Hirschfeld (1989). Several researchers, most recently Babai et al. (1991), have since improved the result by showing that the same conclusion follows from weaker hypotheses.

A detailed discussion of probabilistic complexity classes is given in Johnson (1990) and Zachos (1988).

4.3 PROBABILISTIC ALGORITHMS IN NUMBER THEORY: FACTORING AND PRIMALITY TESTING

The **primality testing problem** is that of determining whether an integer *N* is prime or composite, and the **factoring problem** is that of finding all the prime factors of *N*. These are two of the most basic computational problems in number theory.

The primality testing and factoring problems have the added practical significance of playing complementary roles in the RSA cryptosystem, which is the current best candidate for public key cryptography and for digital signatures. The practicality of constructing RSA ciphers depends on primality testing's being easy to do, whereas the security of the cipher depends on the apparent difficulty of factoring large numbers.

At present, the fastest known fully analyzed algorithms for both primality testing and factoring are probabilistic.

Theoretical and practical progress in primality testing has been rapid since 1977. Analysis of various primality testing algorithms led to study of the complexity classes described in §4.2.

Most methods of primality testing are based on variants of Fermat's little theorem, which asserts that if *N* is prime, then

$$a^{N-1} \equiv 1 \pmod{N} \qquad (4.1)$$

if *N* does not divide *a*. Most composite numbers have many residues *a* (mod *N*) for which $a^{N-1} \not\equiv 1$ (mod *N*); hence a test for probable primality is to test whether $a^{N-1} \equiv 1$ (mod *N*) for many random *a*. Unfortunately, there exist composite numbers *N* for which $a^{N-1} \equiv 1$ (mod *N*) if (*a, N*) = 1; these are called **pseudoprimes**. Refinements of the condition (4.1) can be used to rule out pseudoprimes, either probabilistically or deterministically.

The primality testing problem is believed by many to be solvable in polynomial time. In fact, Miller (1976) proposed a deterministic algorithm believed to have this property:

Miller's Primality Test

```
Given $N \equiv 1$ (mod 2) find the largest $2^k$ exactly dividing N - 1.
For $1 \leq a \leq 10$ $(\log N)^2$ do:
    Is $a^{N-1} \equiv 1$ (mod N)? If not, output "N composite."
    For $1 \leq j \leq k$ do
        Is $a^{N-1/2^j} \not\equiv \pm 1$ (mod N)? If so, output "N composite."
        Is $a^{N-1/2^j} \equiv -1$ (mod N)? If so, exit loop.
    od
od.
Output "N prime."
```

Note that a^{N-1} (mod *N*) may be computed in polynomial time by expanding *N* in binary and computing a^{2^j} (mod *N*) by repeated squaring and reduction (mod *N*), so this is a polynomial-time algorithm. Miller proved that this algorithm is always correct when it outputs "*N* composite" and that, assuming the truth of the extended Riemann hypothesis (ERH), it is correct when it outputs "*N* prime." However, because the ERH is unproven, it is currently not actually known to be a primality test.

Probabilistic analogs of this algorithm were developed by Solovay and Strassen (1977) and Rabin (1980):

Solovay-Strassen Primality Test

```
Given $N \equiv 1$ (mod 2) find the largest $2^k$ exactly dividing N - 1.
Draw a at random from [1, N - 1]
Is $a^{N-1} \equiv 1$ (mod N) ? If not, output "N composite".
For $1 \leq j \leq k$ do
    Is $a^{N-1/2^j} \not\equiv \pm 1$ (mod N)? If so, output "N composite."
    Is $a^{N-1/2^j} \equiv -1$ (mod N)? If so, exit loop.
od.
Output "N prime".
```

They proved that, if *N* is composite, this test outputs "*N* is composite" with probability at least 3/4 and, if *N* is prime, it always outputs "*N* is prime." This shows that $\mathit{Primes} \equiv \mathcal{L} \in \text{co-RP}$. By repeating the test $(\log N)^2$ times, the probability of error can be made exponentially small.

The drawback of the Solovay-Strassen test is that it does not guarantee that *N* is prime when it outputs "*N* prime"; i.e., it provides no proof of primality.

This situation was partially remedied by Goldwasser and Kilian (1986). They gave a probabilistic algorithm that counts points on "random" elliptic curves (mod *N*), and they proved that for almost all primes *p* it finds in probabilistic polynomial time a proof that *p* is prime. However, they could not rule out the possibility of the existence of a small set of "exceptional" *p* for which there was no proof of the kind they searched for. Recently, Adleman and Huang (1987) announced a proof that $\mathit{Primes} \in \text{RP}$. Their result is technically difficult and proceeds by counting points on "random" Abelian varieties (mod *p*) of certain types. Taken together with the Solovay-Strassen result, it shows that $\mathit{Primes} \in \text{ZPP}$.

In contrast, the fastest fully analyzed deterministic primality testing algorithm is due to Adleman et al. (1983). It tests *N* for primality in time $[O((\log N)^{c_0 \log \log \log N})]$ for a certain positive constant c_0.

For the factoring problem, all known algorithms, whether probabilistic or deterministic, require superpolynomial time. The fastest fully analyzed deterministic algorithm for factoring *N*, due to Pollard (1974), has the exponential running-time bound $O(N^{1/4+\epsilon})$.

The algorithms used in practice for factoring large numbers are probabilistic in nature. The basic approach of most of them is to find solutions to $X^2 \equiv Y^2$ (mod *N*) and hope that $(X + Y, N)$ is a nontrivial factor of *N*. This is done by generating probabilistically many pairs (X_i, a_i) with $X_i^2 \equiv a_i$ (mod *N*) where the a_i's are not squares, and then finding a product of the a_i's that is a square by an auxiliary method. The basic approach is to consider only a_i's having all prime factors less than a bound *y*; such numbers are called ***y*-smooth**. Those a_i's that do not have this property are discarded. Then the *y*-smooth a_i's are factored

$$a_i = \prod_{p_j < y} p_j^{e_{ij}} .$$

Now if sufficiently many such a_i's are known, the matrix $[e_{ij}]$ considered over *GF(2)* must have a linear dependency. The product of the a_i's corresponding to this linear dependency is the desired square Y^2, where X^2 is the product of the corresponding X_i^2 . This approach was first described in the Morrison and Brillhart (1975) continued fraction method. Dixon (1981) gave a fully analyzable version of the method. Let $L = \exp[(\log n \log\log n)^{1/2}]$. The Dixon algorithm uses y-smooth numbers with $y = L$β, for a small constant

Dixon's Random Squares Factoring Algorithm

1. *Test N for primality. If N is not prime, continue.*
2. *Randomly pick* X_i and compute the least positive residue a_i of X_i^2 (*mod N*).
3. *Trial divide* a_i by all primes $p_j < L$β. *If* a_i completely factors as

$$a_i = \prod_{p_j < L^\beta} p_j^{e_{ij}} ,$$

add (X_i, a_i, e_{ij}) *to a list. Continue until the list has* Lβ *-elements* .

4. *The matrix* [e_{ij}] *is now linearly dependent over* GF(2). *Compute a linear dependency, i.e., a set S of rows with*

$$\sum_{i \in S} e_{ij} \equiv 0 \ (mod\ 2) \quad \text{for all} \ j .$$

5. *Set* $X = \prod_{i \in S} X_i$ *and* $Y^2 = \prod_{i \in S} a_i$*. Then* $X^2 \equiv Y^2$ (*mod N*). *Test whether* (*X* + *Y*, *N*) *is a nontrivial factor of N. If so, halt. If not, go to step 2 and repeat.*

Here a dependency in the matrix [e_{ij}] is found using Gaussian elimination over *GF(2)*). Dixon (1981) showed that the probability of finding a factor of a composite *N* in one pass through this algorithm is at least 1/2.

The key to analysis of the algorithm is that the probability that a random integer in [1, *N*] is Lβ-smooth is about $L^{-1/2}$β. Dixon showed that a similar probability estimate holds for a_i that are random quadratic residues X_i^2. Hence the expected time needed to generate the matrix [e_{ij}] is $L^{\beta+\frac{1}{2\beta}}$, and the time needed to Gaussian reduce the matrix [e_{ij}] is about L^3β. This is minimized for β = 1/2, and the algorithm takes time $O(L^{3/2+\epsilon})$ and space $O(L^{1+\epsilon})$ bits. This is Dixon's result.

Note that this expected running time bound,

$$O\left(\exp\left[(\frac{3}{2}+\epsilon)(\log N \ \log\log N)^{1/2}\right]\right) ,$$

is subexponential in *N*; i.e., it represents a superpolynomial speedup of the currently best fully analyzed deterministic factoring algorithm.

Pomerance (1987) gives a refined variant of this probabilistic algorithm that is rigorously proven to run in expected time $O(L^{\sqrt{2}+\epsilon})$. There are several other factoring algorithms of a similar nature that are believed to run in expected time $O(L^{1+\epsilon})$ (see Pomerance, 1982, and Lenstra and Lenstra, 1990) and one such algorithm for which the bound $O(L^{1+\epsilon})$ can be proven rigorously (see Lenstra and Pomerance, 1992).

More recently, a new factoring algorithm, the number field sieve, has been found that incorporates probabilistic ideas and uses arithmetic in an auxiliary algebraic number field to generate $X^2 \equiv Y^2$ (mod *N*). It appears to have running time $O\{\exp[(2+\epsilon)(\log N\,(\log\log N)^2)^{1/3}]\}$ but is likely not to be fully analyzable. It is particularly efficient for factoring numbers of special form, such as x^n +2, and has already been used in practice to factor a 158-digit number; a preliminary description appears in Lenstra et al. (1990).

4.4 COMMUNICATION COMPLEXITY

Communication complexity studies variants of the following problem:

Problem 1 A Boolean function *f* is defined on a finite set *X* × *Y*. Two persons, P_X knowing $x \in X$ and P_Y knowing $y \in Y$, both want to know *f*(*x, y*). How many bits of information do they need to exchange to do this?

This problem was proposed by Yao (1979). It is a basic problem in distributed computation, where the "persons" are processors, each possessing partial information about the current state of the computation.

The **worst-case (deterministic) communication complexity** $\hat{C}_D(f)$ is defined as follows. Let be a deterministic protocol describing a sequence of bit exchanges between P_X and P_Y that always computes *f* correctly. Let *l* (*x, y*) denote the number of bits exchanged using when P_X knows *x* and P_Y knows *y*. The worst-case communication complexity for the protocol is

$$\hat{C}_D(f)_\phi := \max_{(x,y)\in X\times Y} [l_\phi(x,y)],$$

and the worst-case deterministic communication complexity for *f* is

$$\hat{C}_D(f) := \inf_{\phi} [\hat{C}_D(f)_\phi].$$

Randomized algorithms offer provable speedups for some problems in communication complexity. Here one allows protocols in which P_X and P_Y each have access to different sources of independent random bits. Let be a protocol computing *f(x, y)* with probability of error at most ϵ for all (*x, y*). Let $\bar{l}_\phi(x,y)$ denote the expected number of bits needed by when P_X knows *x* and P_Y knows *y*, and define

$$\hat{C}_R(f)_\phi := \max_{(x,y)\in X\times Y} [\bar{l}_\phi(x,y)] .$$

The ϵ**-error randomized communication complexity** is

$$\hat{C}_R(f,\epsilon) := \inf [\hat{C}_R(f)_\phi] ,$$

where the infimum is computed over all protocols with error probability at most ϵ. We single out the **zero-error randomized communication complexity** $\hat{C}_R(f,0)$.

We describe two specific functions for which probabilistic computations give speedups. The first is the **equality function** *equ*. Here *X* = *Y* = {1, 2, ..., *n*}, and

$$equ(x,y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise} . \end{cases}$$

Yao (1979) showed that $\hat{C}_D(equ) \geq \lceil \log(n) \rceil$. In fact, $\hat{C}_D(equ)$ is exactly [log(*n*)] + 1. A protocol attaining this bound is as follows: P_X transmits *x* to P_Y and P_Y transmits back the value *f(x,y)*. Yao (1979) also gave a randomized protocol with error probability at most ϵ and showed that

$$\hat{C}_R(equ)_\phi \leq c_0 \left(\log\log n + \log\left(\frac{1}{\epsilon}\right)\right) ,$$

thus demonstrating a speedup. To describe this protocol, let $\sigma = \sqrt{2}\,\frac{\log n}{\epsilon}$:

Randomized Prime Protocol

1. P_Y picks a random prime p in the interval $\sigma \leq p \leq 2\sigma$. *He transmits p and y* (*mod p*).
2. P_X transmits 1 if $x \equiv y$ (*mod p*) *and* 0 *otherwise.*
3. P_X and P_Y both take the bit transmitted to be the value equ(*x,y*).

If this algorithm outputs *equ*(*x,y*) = 0, this answer is correct, but when it outputs *equ*(*x,y*) = 1, this answer may be incorrect, with probability at most ϵ. For the function *equ*, no speedup exists using a zero-error randomized algorithm (see Orlitsky and El Gamal, 1990).

Next we consider the function *componentwise-equal (c.e.)*, which has $X = Y = \{0,1\}^{n^2}$. Write x = (x_1,..., x_n) with each $x_i \in \{0,1\}^n$ and similarly for y. Then

$$\text{c.e.}(\mathrm{x},\mathrm{y}) = \begin{cases} 1 & \text{if for some } i,\ x_i = y_i \\ 0 & \text{else}. \end{cases}$$

Mehlhorn and Schmidt (1982) showed that $\hat{C}_D(\text{c.e.}) \geq n^2$. They exhibited a zero-error randomized protocol such that

$$\hat{C}_R(\text{c.e.})_0 \leq 5n \log n + c_0 .$$

The protocol is as follows:

Repeated Equality Protocol

1. *For* $1 \leq i \leq n$,

(a) P_X and P_Y determine equ(x_i, y_i) *using the randomized prime protocol with* $\epsilon = (\log n)/n$.

(b) *If they conclude that* $x_i \neq y_i$, *they move to step (a) and next i. If they conclude* $x_i = y_i$, they exchange x_i and y_i and check whether they are actually equal. If so, they output 1 and halt. If not, they move back to step 1.

2. *Output 0 if all i are scanned and no* $x_i = y_i$ are exchanged.

This protocol is zero-error because it checks directly whether $x_i = y_i$ on indices *i* where the randomized prime protocol might have made a mistake. Fürer (1987) further improved the zero-error bound for this function to

$$\hat{C}_R(\text{c.e.}, 0) \leq c_0 n ,$$

for a constant c_0.

Aho et al. (1983) demonstrated that at most a quadratic speedup is possible for zero-error randomized communication complexity relative to worst-case communication complexity; i.e.,

$$\hat{C}_R(f, 0)^2 \geq \hat{C}_D(f) .$$

The second example above shows that this bound is essentially sharp.

Finally, we mention a recent result of Ahlswede and Dueck (1989) that any object among $N = 2^{2^{nR}}$ objects can be identified in blocklength n using a discrete memoryless channel of rate R with arbitrarily small error probability via an encoding procedure that uses randomization.

Orlitsky and El Gamal (1988) give a general survey of communication complexity, and Halstenberg and Reischuk (1990) state the best currently known results about relationships among communication complexity classes.

REFERENCES

Adleman, L., and M.D. Huang (1987), Recognizing primes in random polynomial time, in *Proc. 19th Annual Symposium on Theory of Computing* , ACM Press, New York, 462-469.

Adleman, L., C. Pomerance, and R. Rumely (1983), On distinguishing prime numbers from composite numbers, *Ann. Math.* **117**, 173-206.

Ahlswede, R., and G. Dueck (1989), Identification via channels, *IEEE Trans. Inform. Theory* **35**, 15-29.

Aho, A., J.D. Ullman, and M. Yannakakis (1983), On notions of information transfer in VLSI circuits, in *Proc. 15th Annual Symposium on Theory of Computing*, ACM Press, New York, 133-139.

Babai, L., L. Fortnow, N. Nisan, and A. Wigderson (1991), BPP has subexponential-time simulations unless EXP has publishable proofs, in *Proc. 6th Annual Structure in Complexity Theory Conference*, IEEE Computer Society Press, Los Alamitos, Calif., 213-219.

Boppana, R., and R. Hirschfeld (1989), Pseudorandom generators and complexity classes, in *Randomness and Computation*, S. Micali, ed., Advances in Computing Research, vol. 5, JAI Press, Greenwich, Conn., 1-26.

Dixon, J. (1981), Asymptotically fast factorization of integers, *Math. Comp.* **36**, 255-260.

Füter, M. (1987), The power of randomness for communication complexity, in *Proc. 19th Annual Symposium on Theory of Computing*, ACM Press, New York, 178-181.

Goldwasser, S., and J. Kilian (1986), Almost all primes can be quickly certified, in *Proc. 18th Annual Symposium on Theory of Computing*, ACM Press, New York, 316-329.

Halstenberg, B., and R. Reischuk (1990), Relationships between communication complexity classes, *J. Comput. Syst. Sci.* **41**, 402-429.

Johnson, D.S. (1990), A catalog of complexity classes, in *Handbook of Theoretical Computer Science*, vol. A, J. von Leeuwen et al., eds., North-Holland, New York.

Lenstra, A.K., and H.W. Lenstra, Jr. (1990), Algorithms in number theory, in *Handbook of Theoretical Computer Science*, vol. A, J. von Leeuwen et al., eds., North-Holland, New York.

Lenstra, A.K., H.W. Lenstra, Jr., M.S. Manasse, and J. Pollard (1990), The number field sieve, in *Proc. 22nd Annual Symposium on Theory of Computing*, ACM Press, New York, 564-572.

Lenstra, H.W., and C. Pomerance (1992), A rigorous time bound for factoring integers, *J. Amer. Math. Soc.*, to appear.

Lutz, J. (1990), Pseudorandom sources for BPP, *J. Comput. Syst. Sci* . **41**, 307-320.

Mehlhorn, K., and E.M. Schmidt (1982), Las Vegas is better than determinism in VLSI and distributed computing (extended abstract), in *Proc. 14th Annual Symposium on Theory of Computing*, ACM Press, New York, 330-337.

Miller, G. (1976), Riemann's hypothesis and tests for primality, *J. Comput. Syst. Sci.* **13**, 300-317.

Morrison, M., and J. Brillhart (1975), A method of factoring and the factorization of F_7, *Math. Comp*. **29**, 183-205.

Orlitsky, A., and A. E1 Gamal (1988), Communication complexity, in *Complexity in Information Theory*, Y.S. Abu-Mostafa, ed., Springer-Verlag, New York, 16-61.

Orlitsky, A., and A. E1 Gamal (1990), Average and randomized communication complexity, *IEEE Trans. Inform. Theory* **36**, 3-16.

Pollard, J.M. (1974), Theorems on factorization and primality testing, *Proc. Cambridge Philos. Soc*. **76**, 521-528.

Pomerance, C. (1982), Analysis and comparison of some integer factoring algorithms, in *Computational Methods in Number Theory*, H.W. Lenstra, Jr. and R. Tijdeman, eds., Math. Centre Tract 144, University of Amsterdam, 89-139.

Pomerance, C. (1987), Fast rigorous factorization and discrete logarithm algorithms, in *Discrete Algorithms and Complexity*, D.S. Johnson, T. Nishizeki, A. Nozaki, and H. Wilf, eds., Academic Press, Boston, 119-144.

Rabin, M.O. (1980), Probabilistic algorithms for testing primality, *J. Number Theory* **12**, 128-138.

Solovay, R., and V. Strassen (1977), A fast Monte-Carlo test for primality, *SIAM J. Comput*. **6**, 84-85; erratum **7**, 118.

Yao, A. (1979), Some complexity questions related to distributed computing (extended abstract), in *Proc. 11th Annual Symposium on Theory of Computing*, ACM Press, New York, 209-213.

Yao, A. (1982), Theory and applications of trapdoor functions (extended abstract), in *Proc. 23rd Annual Symposium on Foundations of Computer Science*, IEEE Computer Society Press, Los Alamitos, Calif., 80-91.

Zachos, S. (1988), Probabilistic quantifiers and games, *J. Comput. Syst. Sci*. **36**, 433-451.

5. Probabilistic Algorithms for Defeating Adversaries

Joan Feigenbaum
AT&T Bell Laboratories

ABSTRACT

Randomization appears to be an essential ingredient in algorithms for maintaining some form of privacy. This chapter discusses probabilistic algorithms for authenticating a user and for allowing the private use of shared resources.

5.1 INTRODUCTION

We consider two types of **adversarial computation** in which randomness is used in an essential way.

The first type of scenario concerns **authentication**: Player *A* makes a claim that player *B* may not believe. So there is a need for a protocol in which *A* "proves" to *B* that the claim is correct (or at least provides overwhelming statistical evidence of correctness). Scenarios of this type include:

- *A* is a bank customer, and *B* is an automatic teller machine (ATM). Here, *A*'s "claim" is that he is the legitimate owner of the smart card that he inserts in the ATM.
- *A* is a computing workstation, and *B* is a file server. If there is no direct hardware connection between *A* and *B, A* may send its requests

for files over a telephone line. Of course, anyone can dial the phone number of the file server; *A*'s claim is that it is legitimately entitled to access the files.

- *A* is a television owner, and *B* is a cable company. *A*'s claim is that he is a paying customer and that the broadcast should be descrambled for him.

These are really three examples of the same thing. Player *A* must be able to **authenticate himself** (or **prove his identity**) to *B*. Any potential impersonator A^* must have only a negligible probability of convincing *B* that he is *A*.

Goldwasser et al. (1989) and Babai and Moran (1988) initiated a complexity-theoretic study of such problems. They defined the class of sets with **interactive proof systems** and suggested that this definition correctly captures the notion of a set in which membership of an element can be verified efficiently. A proof system for a set $T \subseteq \{0,1\}^*$ is an interactive protocol between a **prover** *A* (with unlimited computing power) and a **verifier** *B* (limited to probabilistic polynomial-time computation). If If $x \in T$, then *A* is able to convince *B* to accept *x* with high probability. If If $x \notin T$, then no player *A**, even if he deviates from the legitimate behavior of *A*, should be able to convince *B* to accept *x* with more than negligible probability. The more general notion of a **multiprover interactive proof system** was put forth by Ben-Or et al. (1988); in these protocols, the role of the prover is played by several machines $A_1,..., A_m$, each with unlimited computing power. Two ingredients of proof systems are absolutely vital in making the theory interesting: interaction (i.e., the fact that prover(s) and verifier can exchange polynomially many rounds of messages before the verifier decides whether or not to accept), and randomness (i.e., the fact that the verifier can toss coins at any point in the protocol and that cheating provers do not know the outcomes of these tosses at the start of the protocol).

Consider the previous examples, in which *A* is required to prove his identity to *B*. Suppose we use a trivial protocol in which *B* is given a table of users and passwords, *A* is given only his own password, and to prove his identity, *A* simply reveals the password to *B*. (Note the absence of interaction and randomness.) In this scheme, *A* transfers full **knowledge** of his identity proof to *B*. An eavesdropper (or even *B* himself) can turn around and claim to be *A* by presenting the same proof. Is there a proof system that achieves the same goal but transfers no knowledge to *B* except the one bit that *A* is who he claims to be? Goldwasser et al. (1989) define the notion of a **zero-knowledge proof system** formally and propose it as the right way to capture the intuitive requirements of an authentication scheme. Basically, an interactive proof system for the set *T* is zero-knowledge if, for any $x \in T$, anything that a (potentially cheating) verifier B^* can compute given the transcript of his interaction with *A*, he can also compute given only the one bit of information "*x* is in *T*." Zero-knowledge is defined analogously for multiprover interactive proof systems.

The second type of scenario concerns **private use of shared resources** . A community of users shares a valuable computing resource, and each member of the community would like to use the resource without revealing his private data. Scenarios of this type include the following:

- The users are scientists, and the resource is a government supercomputer.
- The users are pairs of computers or telephones, and the shared resource is a communication line.
- The users are potential customers, and the resource is the vendor's proprietary software.
- The users are investors, and the resource is a service that analyzes financial markets.

Abadi et al. (1989) formalized this scenario as follows. There is a function *f* that is intractable for the user *A* to compute, because *A* has limited computing power. The shared resource *B* is a program or device that computes *f*. (More generally, *B* can compute some function *g* that is closely related to *f*.) *A* has a private input *x*; he would like to use this resource to find out *f*(*x*) without revealing *x* to *B*.

An **instance-hiding scheme** for the function *f* is a pair of functions *E* and *D* with the following properties. The inputs to *E* are *A*'s private data *x* and some random bits *r*. *A* computes $y = E(x, r)$ and sends it to *B*, who sends back *f*(*y*). Then *A* computes *D*(*x,r,f*(*y*)), which is the desired answer *f*(*x*). *B* should not be able to deduce *x* from *y*. *A*, on the other hand, can deduce *f*(*x*) from *f*(*y*), because he has the random bits *r* that were used to compute *y* from *x*. (More generally, *B* could send back *g*(*y*), and *A* could use it to compute *f*(*x*).) Note that such a scheme makes sense only if the functions *E* and *D* are easier to compute than the function *f*.

Randomness plays an essential role here, just as it does in zero-knowledge. The input y that is sent to B is a random variable. If x_1 and x_2 are both possibilities, from B's point of view, for A's private input, and R is uniformly distributed over all bit-strings of the appropriate length, then $Y_1 = E(x_1, R)$ and $Y_2 = E(x_2, R)$ should be identically distributed. Hence B learns nothing about x when he receives a random query.

In the next section, we give examples of zero-knowledge proof systems and instance-hiding schemes. In §5.3, we state some of the major known results on both types of schemes. Section 5.3 also contains some suggestions for further work.

5.2 EXAMPLES

5.2.1 Authentication

Feige et al. (1988) developed the following protocol to allow a community of users to authenticate themselves to each other. For each user, there is a pair (I, S). The public information I can be interpreted as the user's identity and the private information S as his secret key. The goal of the authentication scheme is to allow a user with identity I to convince another user that he "knows" the corresponding key S without revealing anything about S beyond the fact that he knows it. (In the language of cryptography theory, user I provides a **zero-knowledge proof of knowledge** of the key S.) The effectiveness of the scheme is based on the assumption that it is computationally infeasible to compute square roots modulo a large composite integer with unknown factorization; this is provably equivalent to the assumption that factoring large integers is difficult.

Modulus Generation. A **trusted center** generates two large primes, each congruent to 3 mod 4. The product m of these primes is published, but the primes are not.

Note that -1 is a quadratic nonresidue modulo m: that is, there is no a such that a^2 = -1 mod m. In what follows, $\mathbf{Z}_m^*[+1]$ denotes the set of integers between 1 and m that are relatively prime to m and have Jacobi symbol +1 with respect to m.

Key Generation. Each user chooses t_1 random numbers $S_1, \ldots, S_{t_1}$ in $\mathbf{Z}_m^*[+1]$ and t_1 random bits $b_1, \ldots, b_{t_1}$. He sets I_j equal to $(-1)^{b_j}/S_j^2$ mod m, for $1 \leq j \leq t_1$. This user's identity, which he publishes, is $I = (I_1, \ldots, I_{t_1})$, and his secret key, which he keeps private, is $S = (S_1, \ldots, S_{t_1})$.

Proofs of Identity. User *A* authenticates himself to user *B* as follows: Let *I* be *A*'s published identity and *S* his secret key. *A* and *B* repeat the following four steps t_2 times:

1. *A* picks a random *R* in $\mathbf{Z}_m^*[+1]$ and a random bit *c* and sends $X = (-1)^c R^2$ mod *m* to *B*.
2. *B* sends a random vector of bits $(E_1, \ldots, E_{t_1})$ to *A*.
3. *A* sends $Y = R \cdot \prod_{E_j=1} S_j$ mod *m* to *B*.
4. *B* verifies that $Y^2 \cdot \prod_{E_j} I_j$ mod *m* is equal to ±*X*.

B believes that *A* is who he claims he is if the verification in step (4) succeeds in each of the t_2 trials.

Feige et al. (1988) show that this scheme works correctly for the values $t_1 = O(\log \log m)$ and $t_2 =$ (log *m*). The most important feature of the scheme is that there is no need for the prover to have significant computational power. These identity proofs require only a few modular multiplications and can be implemented on smart cards.

5.2.2 Computing with Encrypted Data

In our first example of computation with encrypted data, the hard-to-compute function is the discrete logarithm modulo primes. Let *p* be a large prime and *g* be a generator for the multiplicative group $\mathbf{Z}_p^*$. If $x = g^e$ mod *p*, where *e* is an element of {1,..., *p*-1}, then *e* is called the **discrete logarithm of** *x* (with respect to *g* and *p*). There is an efficient algorithm to compute *x* given *e* that is based on repeated squaring modulo *p*. However, there is no efficient algorithm known to compute *e* given *x*. There are many cryptographic protocols that try to exploit the presumed intractability of the discrete logarithm problem. For example, Diffie and Hellman (1976) give a protocol for **secret key exchange** that is based on the difficulty of discrete logarithms.

Suppose that we have a box *B* that computes discrete logarithms. In complexity theoretic terms, *B* is an **oracle**—we can ask it questions, but we do not know how it works and cannot compute what it computes on our own. In real applications, *B* may be a piece of proprietary software or hardware, or it may be a large table that was computed at great expense. In any case, a discrete logarithm box would be a valuable resource, and many mutually suspicious users would want access to it.

Abadi et al. (1989) give a simple scheme by which *A* can get the discrete logarithm of *x* without revealing *x*. First, *A* chooses an element *r* uniformly at random from {1,..., *p*-1} and computes $y = x\cdot g^r \bmod p$. Next, *A* sends *y* to *B*, and *B* sends back the discrete logarithm of *y*; that is, *B* sends *e'* such that $y = g^{e'} \bmod p$. Finally, *A* computes $e = e'-r \bmod p-1$. This exponent *e* is the discrete logarithm of *x*. Because *r* was chosen uniformly at random, *y* is also a uniformly distributed random element of {1,..., *p*-1}. Hence, no one else who has access to the box *B* and overhears *A*'s query gets any information about *A*'s private input *x*.

In our second example, the hard function is a multivariate polynomial $h \in K[X_1, \ldots, X_n]$, where *K* is a finite field. Such a polynomial may be hard to compute, because the number of terms may be exponential in *n*. Let *d* be the degree of *h*. Assume that $|K| > d + 1$ and that $\alpha_1,..., \alpha d+1$ are distinct nonzero elements of *K*. The following scheme, in which user *A* obtains $h(x_1,..., x_n)$ without revealing $x = (x_1,..., x_n)$, was devised by Beaver and Feigenbaum (1990) and Lipton (1991).

Once again, let *B* be a box that computes *h*. For this example, we must have *d* + 1 copies, say $B_1,..., B_{d+1}$, of *B*; furthermore, for $i \neq j$, the communication between *A* and B_i cannot be overheard by B_j. Think of the B_i's as distinct physical boxes that are kept in *d* + 1 different locations.

Let $c = (c_1,..., c_n)$ be an *n*-tuple of elements of *K*, and let *Z* be an indeterminate that is distinct from each of $X_1, ..., X_n$. Consider the univariate polynomial

$$H(Z) \equiv h(c_1 Z + x_1, \ldots, c_n Z + x_n).$$

Note that *H* has degree at most *d* and that

$$H(0) = h(x_1, \ldots, x_n).$$

To compute *h*(*x*) privately, *A* first chooses *c* uniformly at random from K^n. Next, *A* computes $y_1, ..., y_{d+1}$, where

$$y_j \equiv (c_1\alpha_j + x_1, \ldots, c_n\alpha_j + x_n).$$

For $1 \leq j \leq d+1$, *A* sends the query y_j to B_j and gets back $h(y_j)$. Note that $h(y_j) = H(\alpha_j)$. Thus, after receiving these answers, *A* has *d*+1 distinct points $\{(\alpha_j, H(\alpha_j))\}$ on the degree-*d* univariate polynomial *H*. Using these points, *A* can recover *H* by interpolation; the desired answer *h*(*x*) is just the constant term of *H*.

Each query y_j is a uniformly distributed random element of K^n. Of course, y_i and y_j are correlated; if B_i and B_j could communicate, they could recover *x*. In isolation, however, B_j and its other users learn nothing about *A*'s private input *x* from the random query y_j.

5.3 ZERO-KNOWLEDGE PROOF SYSTEMS AND INSTANCE-HIDING SCHEMES

In this section, we give a brief, informal summary of the known results on zero-knowledge proof systems and instance-hiding schemes. More extensive summaries, as well as more precise formulations of the concepts, are given by Feigenbaum (1992) and Brassard (1990).

The first basic problem is to characterize the sets that have interactive proof systems. A complete characterization was achieved in a sequence of papers culminating in the work of Lund et al. (1990) and Shamir (1990).

Theorem 5.1 *The sets with interactive proof systems are exactly those that are recognizable in polynomial space.*

The second basic question is whether every set that has an interactive proof system in fact has one that is zero-knowledge. Recall that a function is **one-way** if it is easy to compute but hard to invert. The next result follows from the work of Impagliazzo and Yung (1988) and Naor (1991).

Theorem 5.2 *Assume that one-way functions exist. Then every set that has an interactive proof system has one that is zero-knowledge.*

In a forthcoming paper, Ostrovsky and Wigderson (1992) show that if there is a zero-knowledge proof system for any set that is hard on average, then there is a one-way function.

The set-recognition power of multiprover interactive proof systems has also been completely characterized; this result is due to Babai et al. (1991). The question of zero-knowledge also is settled completely for multiprover interactive proof systems; see Ben-Or et al. (1988).

Theorem 5.3 *The sets with multiprover interactive proof systems are exactly those recognizable in nondeterministic exponential time. Furthermore, all of these sets have multiprover systems that are zero-knowledge*.

In the instance-hiding scheme for the discrete logarithm function that is given in §5.2.2, player *A* queries only one box *B*. This is an example of a **one-oracle instance-hiding scheme**. The scheme for the multivariate polynomial is an example of a **multi-oracle instance-hiding scheme**.

The first basic question here is to determine exactly which sets have instance-hiding schemes that leak no information about *x* to the oracles. Actually, it impossible to find schemes that leak *absolutely* no information—*A* must leak to *B* at least the length of *x* if the function computed by *B* is nontrivial. (This is stated and proven precisely in Abadi et al., 1989.) Thus, the real question is which sets have instance-hiding schemes that leak at most the length of *x*.

For one-oracle schemes, Abadi et al. (1989) obtained the following conditional negative result:

Theorem 5.4 *No NP-hard set has a one-oracle instance-hiding scheme that leaks at most the length of x, unless the polynomial-time hierarchy collapses at the third level.*[1]

Beaver and Feigenbaum (1990) used the **low-degree polynomial trick** given in §5.2.2 to obtain a universal construction of multi-oracle instance-hiding schemes:

Theorem 5.5 *Every Boolean function of n bits has an* (*n* + 1)-*oracle instance-hiding scheme that leaks at most the length of x to each oracle*.

Interestingly, this low-degree polynomial trick, which was devised in order to construct instance-hiding schemes, became a crucial ingredient in the characterization of the set-recognition power of interactive proof systems, both one-prover and multiprover. A thorough explanation of this connection appears in Brassard (1990).

[1] The zero'th level of the hierarchy is polynomial time, and the first is nondeterministic polynomial time. These are the familiar complexity classes P and NP. The conjectured inequality $P \neq NP$ can also be stated as "the polynomial-time hierarchy does not collapse at the zero'th level." The second, third, and higher levels of the hierarchy are further generalizations of the notion of polynomial-time computation. The conjecture that $P \neq NP$ can be generalized to "the polynomial-time hierarchy does not collapse at the *i*th level," for any given value of *i*.

Finally, we remark that it is natural to consider **zero-knowledge, instance-hiding proof systems**. These are proof systems in which the verifier does not learn the proof, and the prover does not learn what he is proving! Beaver et al. (1991) formalize this notion and prove the following positive result:

Theorem 5.6 *Every set recognizable in nondeterministic exponential time has a multiprover interactive proof system that is both instance-hiding and zero-knowledge*.

One theoretical open question remaining concerns the power needed by a prover; for example, how powerful must the prover be in a proof system for a co-NP-complete set? For a list of other open theoretical questions, see Feigenbaum (1992).

There are many open questions concerning the applicability of this theory to practical computing. The authentication protocol of §5.2.1 forms the basis of a working, commercial system that is in wide use today. We do not know any other examples of working systems that use the theory. This immediately suggests two questions.

First, the proof system of Feige et al. (1988) is based on a highly structured, number-theoretic problem that does not have interesting complexity-theoretic properties. None of the elegant general results stated above are relevant to this one known example of a practical proof system. Can the general results of the theory be applied in practice? For example, can zero-knowledge proof systems for NP-complete problems be used for authentication?

Second, what other practical applications exist for zero-knowledge proof systems? Many ambitious proposals have been made; see, for example, the work of Goldreich et al. (1987). These proposals have enhanced the theoretical literature in cryptography and probably can be put to use in real systems; so far, however, we know of no such uses.

Most of the existing work on instance-hiding is currently not practical. What is the right application domain in which one can make practical use of these ideas?

REFERENCES

Abadi, M., J. Feigenbaum, and J. Kilian (1989), On hiding information from an oracle, *J. Comput. Syst. Sci*. **39**, 21-50.

Babai, L., and S. Moran (1988), Arthur-Merlin games: A randomized proof system and a hierarchy of complexity classes, *J. Comput. Syst. Sci*. **36**, 254-276.

Babai, L., L. Fortnow, and C. Lund (1991), Nondeterministic exponential time has two-prover interactive protocols, *Comput. Complexity* **1**, 3-40.

Beaver, D., and J. Feigenbaum (1990), Hiding instances in multioracle queries, in *Proc. 7th Annual Symposium on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science no. 415, Springer-Verlag, New York, 37-48.

Beaver, D., J. Feigenbaum, and V. Shoup (1991), Hiding instances in zero-knowledge proof systems, in *Proc. 10th CRYPTO*, Lecture Notes in Computer Science no. 537, Springer-Verlag, New York, 326-338.

Ben-Or, M., S. Goldwasser, J. Kilian, and A. Wigderson (1988), Multiprover interactive proof systems: How to remove intractability assumptions, in *Proc. 20th Annual Symposium on Theory of Computing*, ACM Press, New York, 113-131.

Brassard, G. (1990), Cryptology Column #4: Hiding information from oracles, *SIGACT News* **21**, Spring issue, Association for Computing Machinery, 5-11.

Diffie, W., and M. Hellman (1976), New directions in cryptography, *IEEE Trans. Inform. Theory* **22**, 644-654.

Feige, U., A. Fiat, and A. Shamir (1988), Zero knowledge proofs of identity, *J. Cryptology* **1**, 77-94.

Feigenbaum, J. (1992), Overview of interactive proof systems and zero knowledge, in *Contemporary Cryptology: The Science of Information Integrity*, G. Simmons, ed., IEEE Press, New York, 423-440.

Goldreich, O., S. Micali, and A. Wigderson (1987), How to play ANY mental game, in *Proc. 19th Annual Symposium on Theory of Computing* , ACM Press, New York, 218-229.

Goldwasser, S., S. Micali, and C. Rackoff (1989), The knowledge complexity of interactive proof systems, *SIAM J. Comput*. **18**, 186-208.

Impagliazzo, R., and M. Yung (1988), Direct minimum-knowledge computations, in *Proc. 7th CRYPTO*, Lecture Notes in Computer Science no. 293, Springer-Verlag, New York, 40-51.

Lipton, R. (1991), New directions in testing, in *Distributed Computing and Cryptography*, DIMACS Series on Discrete Mathematics and Theoretical Computer Science, vol. 2, American Mathematical Society, Providence, R.I., 191-202.

Lurid, C., L. Fortnow, H. Karloff, and N. Nisan (1990), Algebraic methods for interactive proof systems, in *Proc. 31st Annual Symposium on Foundations of Computer Science*, IEEE Computer Science Press, Los Alamitos, Calif., 2-10.

Naor, M. (1991), Bit commitment using pseudo-randomness, *J. Cryptology* **4**, 151-158.

Ostrovsky, R., and A. Wigderson (1992), One-way functions are essential for non-trivial zero-knowledge, submitted to 33rd Annual Symposium on Foundations of Computer Science (IEEE).

Shamir, A. (1990), IP = PSPACE, in *Proc. 31st Annual Symposium on Foundations of Computer Science*, IEEE Computer Science Press, Los Alamitos, Calif., 11-15.

6. Pseudorandom Numbers

Jeffrey C. Lagarias
AT&T Bell Laboratories

ABSTRACT

This chapter surveys the problem of generating pseudorandom numbers. It lists many of the known constructions of pseudorandom bits. It outlines the subject of computational information theory. In this theory the fundamental object is a secure pseudorandom bit generator. Such generators are not theoretically proved to exist, although functions are known that appear to possess the required properties. In any case, pseudorandom number generators are known that work reasonably well in practice.

6.1 INTRODUCTION

A basic ingredient needed in any algorithm using randomization is a source of "random" bits. In practice, in probabilistic algorithms or Monte Carlo simulations, one uses instead "random-looking" bits. A **pseudorandom bit generator** is a deterministic method to produce from a small set of "random" bits called the **seed** a larger set of random-looking bits called pseudorandom bits.

There are several reasons for using pseudorandom bits. First, truly random bits are hard to come by. Physical sources of supposedly random bits, which rely on chaotic, dissipative processes such as varactor diodes, typically produce correlated bits rather than independent sequences of bits. Such sources also produce bits rather slowly (see Schuster, 1988). Hence one would like to conserve the number of random bits needed in a computation. Second, the deterministic character of pseudorandom bit sequences permits the easy reproducibility of computations. A third reason arises from cryptography: the existence of secure pseudorandom bit generators is essentially equivalent to the existence of secure private-key cryptosystems.

For Monte Carlo simulations, one often wants **pseudorandom numbers**, which are numbers simulating either independent draws from a fixed probability distribution on the real line **R**, or more generally numbers simulating samples from a stationary random process. It is possible to simulate samples of any reasonable distribution using as input a sequence of i.i.d. 0-1 valued random variables (see Devroye, 1986, and Knuth and Yao, 1976). Hence the problem of constructing pseudorandom numbers is in principle reducible to that of constructing pseudorandom bits.

Section 6.2 describes explicitly some pseudorandom bit generators, as well as some general principles for constructing pseudorandom bit generators. Most of these generators have underlying group-theoretic or number-theoretic structure.

Section 6.3 describes the subject of computational information theory and indicates its connection to cryptography. The basic object in this theory is the concept of a secure pseudorandom bit generator, which was proposed by Blum and Micali (1982) and Yao (1982). The basic properties characterizing a secure pseudorandom bit generator are "randomness-increasing" and "computationally unpredictable." Recently obtained results are that if one of the following objects exist then they all exist:

1. a secure pseudorandom bit generator,
2. a one-way function, and
3. a secure (block-type) private-key cryptosystem.

The central unsolved question is whether any of these objects exist. However, good candidates are known for one-way functions. A major difficulty in settling the existence problem for this theory is summarized in the following heuristic:

Unpredictability Paradox. *If a deterministic function is unpredictable, then it is difficult to prove anything about it; in particular, it is difficult to prove that it is unpredictable*.

Section 6.4 surveys results on the statistical performance of various known pseudorandom number generators. In particular, a pseudorandom bit generator called the RSA generator produces secure pseudorandom bits (in the sense of §6.2) if the problem of factoring certain integers is computationally difficult. A good general reference for pseudorandom number generators is Knuth (1981, Chapter 3).

6.2 EXPLICIT CONSTRUCTIONS OF PSEUDORANDOM BIT GENERATORS

Where might pseudorandom number generators come from? There are several general principles useful in constructing deterministic sequences exhibiting quasi-random behavior: expansiveness, nonlinearity, and computational complexity.

Expansiveness is a concept arising from dynamical systems. A flow f_t on a metric space is **expansive** (or **hyperbolic**) at a point $x \in M$ if nearby trajectories diverge (locally) at an exponential rate from each other, i.e., $|f_t(x) - f_t(y)| \geq \lambda^t |x - y|$ provided $|x\text{-}y| < \delta$, where $\lambda > 1$ and for $0 \leq t < t_0$. Such flows exhibit sensitive dependence on initial conditions and are numerically unstable to compute. A discrete version of this phenomenon for maps on the interval $I = [0, 1]$ is a map $T: I \to I$ such that $|T'(x)| > 1$ for almost all all $x \in [0, 1]$; e.g., $T(x) = \; x$ (mod 1), where $\; > 1$. In particular, it requires knowledge of at least the first $O(n)$ bits of $T(n)(x)$ to determine if $x \in [0, \frac{1}{2})$ or $x \in [\frac{1}{2}, 1)$ as $n \to \infty$.

Nonlinearity provides a second source of deterministic randomness. Functional compositions of linear maps are themselves linear, but nonlinear polynomial maps, when composed, can yield exponentially growing nonlinearity; e.g., if $f(x) = x^2$ and $f(j)= f(f(j\text{-}1)(x))$, then $f^{(n)}(x) = x^{2^n}$. Even the simple nonlinear map $f(x) = ax(1 - x)$ for $a \in [0, 4]$, which maps [0,1] into [0,1], exhibits extremely complicated dynamics under iteration (see Collet and Eckmann, 1980).

Computational complexity is a third source of deterministic randomness. Kolmogorov (1965), Chaitin (1966), and Martin-Lof (1966) defined a finite string $A = a_1 \ldots a_k$ of bits to be **Kolmogorov-random** if the length of the shortest input to a fixed universal Turing machine $\mathcal{T}$ that halts and outputs A is of length greater than or equal to kc_0 for a constant c_0. This notion of randomness is that of **computational incompressibility**. Unfortunately, it is an undecidable problem to tell whether a given string A is Kolmogorov random. However, one obtains weaker notions of computational incompressibility by restricting the amount of computation allowed in attempting to compress the string *A*. For example, given a nondecreasing time-counting function *T* such as $T(x) = 5x^3$, a string *A* is ***T*-incompressible** if the shortest input to $\mathcal{T}$ that halts in $T(|A|)$ steps and outputs *A* has length greater than $|A|$. This notion of incompressibility is effectively computable. One can similarly define incompressibility of ensembles of strings *S* with respect to time complexity classes such as PTIME. In fact, the accepting function for membership in a "hard" set in a time-complexity class $\mathcal{T}$ behaves "randomly" when viewed as input to a Turing machine that has a more restricted amount of time available. This idea goes back at least to Meyer and McCreight (1971).

The three principles all involve some notion of **functional composition** or of iteration of a function. Indeed, running a computation of a Turing machine can be viewed as a kind of functional composition with feedback. The first two principles directly lead to candidates for number-theoretic pseudorandom number generators (see below). The computational complexity principle leads to pseudorandom number generators having provably good properties with respect to some level of computational resources, but most of these are not of use in practice because one must use more than the allowed bound on computation to find them in the first place. Finally, we note that there are very close relations between expansiveness properties of mappings and the Kolmogorov-complexity of descriptions of their trajectories (see Brudno, 1982).

A variety of functions have been proposed for use in pseudorandom bit generators. We give several examples below. Most of these functions are number-theoretic, and nearly all of them have an underlying group structure.

Example 1 (Multiplicative Congruential Generator). The generator is defined by

$$x_{n+1} \equiv a x_n + b \pmod{M},$$

where $0 \le x_n \le M-1$. Here (*a, b, M*) are the parameters describing the generator and x_0 is the seed. This was one of the first proposed pseudorandom number generators. Generators of this form are widely used in practice in Monte Carlo methods, taking x_i/M to simulate uniform draws on [0,1] (see Knuth, 1981; Rubinstein, 1982; and Marsaglia, 1968). More generally, one can consider polynomial recurrences (mod *N*).

Example 2 (Power Generator). The generator is defined by

$$x_{n+1} \equiv (x_n)^d \pmod{N}.$$

Here (*d, N*) are parameters describing the generator and x_0 is the seed.

An important special case of the power generator occurs when $N = p_1p_2$ is a product of two distinct odd primes. The first case occurs when (*d*,   (*N*)) = 1, where

$$\phi(N) = \#(\mathbf{Z}/N\mathbf{Z})^* = (p_1 - 1)(p_2 - 1)$$

is Euler's totient function that counts the number of multiplicatively invertible elements (mod *N*). Then the map $x \to x^d$ (mod *N*) is one-to-one on (***Z/NZ***)*, and this operation is the encryption operation of the RSA public-key cryptosystem (see Rivest et al., 1978), where (*d, N*) are publicly known. We call this special case an **RSA generator**.

Example 3 (Discrete Exponential Generator). The generator is defined by

$$x_{n+1} \equiv g^{x_n} \pmod{N}.$$

Here (*g, N*) are parameters describing the generator and x_0 is the seed.

A special case of importance occurs when *N* is an odd prime *p* and *g* is a primitive root (mod *p*). Then the problem of recovering x_n given (x_{n+1}, *g, P*) is the **discrete logarithm problem**, an apparently hard number-theoretic problem (see Odlyzko, 1985). The discrete exponentiation operation (mod *p*) was suggested for cryptographic use in the key exchange scheme of Diffie and Hellman (1976). A **key exchange scheme** is a method for two parties to agree on a secret key using an insecure channel. Blum and Micali (1982) gave a method for generating a set of pseudorandom bits whose security rests on the difficulty of solving the discrete logarithm problem.

Example 4 (Kneading Map). Consider a bivariate transformation

$$(x_{n+1}, y_{n+1}) := (y_n, x_n + f(y_n, z_n)),$$

where *f* is a fixed bivariate function, usually taken to be nonlinear. The function *f*(·,·) determines the generator, and (x_0, y_0) and the family {z_n}

constitute the seed. One often takes $z_n := K$ for all integers n, for a fixed K. This mapping has the feature that it is one-to-one, with inverse

$$(x_n, y_n) := (y_{n+1} - f(x_{n+1}, z_n), x_{n+1}).$$

One can generalize this construction to take x, y, and $\mathbf{f}(\cdot, \cdot)$ to be vector-valued. The **data encryption standard (DES)** cryptosystem is composed of 16 iterations of (vector-valued) maps of this type, where (x_0, y_0) are the data to be encrypted, all $z_i = K$ compose to key, and f is a specific nonlinear function representable as a polynomial in several variables.

Other examples include the $1/P$-generator and the square generator studied in Blum et al. (1986) and cellular automata generators proposed by Wolfram (1956).

In addition to these examples, more complicated pseudorandom number generators can be built out of them using the following two mixing constructions:

Construction 1 (Hashing). If $\{x_n\}$ are binary strings of k bits and $H: \{0,1\}^k \to \{0,1\}^l$ with $l < k$ is a fixed function, called in this context a hash function, then define $\{z_n\}$ by

$$z_n = H(x_n).$$

The traditional role of hash functions is for data compression, in which case l is taken much smaller than k. Sets of good hash functions play an important role in the theory of pseudorandom generators, where they are used to convert highly nonuniform probability distributions on k inputs into nearly uniform distributions on l inputs; see, for instance, Goldreich et al. (1988).

A very special case of this construction is the **truncation operator**

$$T_{j,l}(x) = \lfloor 2^{-j} x \rfloor \pmod{2^l},$$

which extracts from a k-bit string x the substring of l bits starting j bits from its right end. In its most extreme form, $T(x) \equiv x$ (mod 2) extracts the least significant bit. The truncation operation was originally suggested by Knuth to be applied to linear congruential generators to cut off half their bits (saving the high-order bits) as a way of increasing apparent randomness. This idea was used in a number of schemes for encrypting passwords to computer systems. However, it is insecure. Reeds (1979) successfully cryptanalyzed one such system.

Construction 2 (Composition). Let $\{x_n\}$ and $\{y_n\}$ be sequences of k-bit strings, let $* : \{0,1\}^k \times \{0,1\}^k \to \{0,1\}^k$ be a binary operation, and set

$$z_n = x_n * y_n .$$

The simplest case occurs with $k = 1$, where * is the XOR operation:

$$z_n := x_n + y_n \pmod 2 ,$$

where z_n, x_n, and y_n are viewed as k-dimensional vectors over *GF(2)* . More generally, whenever the operation table of * on $\{0,1)^k \times \{0,1\}^k$ forms a Latin square (but is not necessarily either commutative or associative), then the * operation has certain randomness-increasing properties when the elements $\{x_n\}$and $\{y_n\}$ are independent draws from fixed distributions Q and Q on $\{0, 1\}^k$ (see Marsaglia, 1985).

There is a constant search for new and better practical pseudorandom number generators. Recently, Marsaglia and Zaman (1991) proposed some simple generators that have extremely long periods; whether they have good statistical properties remains to be determined.

6.3 COMPUTATIONAL INFORMATION THEORY AND CRYPTOGRAPHY

In the last few years, there has been extensive development of a theoretical basis of secure pseudorandom number generators. This area is called **computational information theory**.

Computational information theory represents a blending of information theory and computational complexity theory (see Yao, 1988). The basic notion taken from information theory is **entropy**, a measure of information. Here, "information" equals "randomness." Information theory was developed by Shannon in the 1940s, and a relationship between computational complexity and information theory was first observed by Kolmogorov. Both Shannon and Kolmogorov assumed in defining "amount of information" that unbounded computational resources were available to extract that information from the data. Computational information theory assumes that the amount of computational resources is bounded by a polynomial in the size of the input data.

Here we indicate the ideas behind the theory, assuming some familiarity with the basic concepts of computational complexity theory for uniform models of computation (Turing machines) and for nonuniform models of computation (Boolean circuits). Turing machines and polynomial-time computability are discussed in Garey and Johnson (1979), and circuit complexity is described in Savage (1976) and Karp and Lipton (1982).

What is a pseudorandom number generator? The basic idea of pseudo-randomness is to take a small amount of randomness described by a seed drawn from a given source probability distribution and to produce deterministically from the seed a larger number of apparently random bits that simulate a sample drawn from another target probability distribution on a range space. That is, it is **randomness-increasing**.

The amount of randomness in a probability distribution is measured by its **binary entropy** (or **information**), which for a discrete probability distribution *P* is

$$H(P) = -\sum_{x} p(x) \log_2 p(x),$$

where *x* runs over the atoms of *P*. In particular

$$H(U_k) = k;$$

i.e., "*k* coin flips give *k* bits of information." The notion of randomness-increasing is impossible in classical information theory because *any deterministic mapping G applied to a discrete probability distribution P never increases entropy*; i.e.,

$$H(G(P)) \leq H(P).$$

However, this may be possible when computing power is limited. Indeed, what may happen is that *G*(*P*) may approximate a target distribution *Q* having a much higher entropy so well that, within the limits of computing power available, one cannot tell the distributions *G*(*P*) and *Q* apart. If *H*(*Q*) is much larger than *H*(*P*), then we can say *G* is **computationally randomness-increasing**.

The fundamental principle is *a definition of pseudorandom numbers is always relative to the use to which the pseudorandom numbers are to be put*. This use is to simulate the target probability distribution to within a specified degree of approximation.

We measure the degree of approximation using statistical tests. Let *P* be the source probability distribution, *G* (*P*) the probability distribution generated by the pseudorandom number generator *G*, and *Q* the target probability distribution to be approximated by $G(P)$. A **statistic** is any deterministic function $\sigma(x)$ of a sample x drawn from a distribution, and a **statistical test** σ consists of computation of a statistic σ drawn from $G(P)$. We say that distributions P_1 and P_2 on the same sample space $\mathcal{S}$ are **ϵ-indistinguishable using the statistic** σ on $\mathcal{S}$ provided that the expected values of $\sigma(x)$ drawn from P_1 and P_2, respectively, agree to within the tolerance level ϵ; i.e.,

$$|E[\sigma(x) : x \in P_1] - E[\sigma(x) : x \in P_2]| \le \epsilon .$$

Given a collection $\mathcal{T} = \{(\sigma_i, \epsilon_i)\}$ of statistical tests σ_i with corresponding tolerance levels ϵ_i, we say that G is a **$\mathcal{T}$-pseudorandom number generator** from source P to target Q provided that $G(P)$ is ϵ_i-indistinguishable from Q for all the statistical tests σ_i drawn from $\mathcal{T}$.

To make these ideas clearer, we allow as source distribution the uniform distribution U_k on the set $\{0, 1\}^k$ of binary strings of length k, and for target distributions we allow either U_ℓ for some ℓ, or else $U([0,1]^\ell)$, which is the distribution of ℓ independent draws from the uniform distribution on $[0, 1]$.

As an example, consider the linear congruential generator $x_{n+1} \equiv a x_n + b$ (mod M) with $M = 2^k$ -1, where the seed x_0 is drawn from $0 \le x_0 \le 2^k - 1$ and is viewed as an element of $\{0, 1\}^k$ drawn with distribution U_k. We use this generator to produce a vector of ℓ iterates, $G(x_0) = (x_1/M, x_2/M, \ldots, x_\ell/M)$, and view $G(U_k)$ as approximating the distribution $U([0,1]^\ell)$. Various statistics that one might consider for $\mathbf{y} = (y_1, \ldots, y_\ell)$ in the sample space $\mathcal{S} = [0,1]^\ell$ are

1. *Average*:

$$\sigma_A(\mathbf{y}) = \frac{1}{\ell} \sum_{i=1}^{\ell} y_i$$

2. *Maximum*:

$$\sigma_M(\mathbf{y}) = \max\{y_i : 1 \le i \le \ell\}$$

3. *Maximal run*:

$$\sigma_R(\mathbf{y}) = \max\{j : y_i < y_{i+1} < \cdots < y_{i+j} \text{ for some } i\}$$

4. *mth autocorrelation*:

$$\sigma_m(\mathbf{y}) = \frac{1}{\ell - m} \sum_{i=1}^{\ell - m} y_i y_{i+m} .$$

It is now a purely mathematical problem to decide what ϵ-tolerance level can be achieved for each of these statistics, viewing $G(U_k)$ as approximating $U([0,1]^\ell)$. Various results for such statistics for linear congruential generators can be found in Niederreiter (1978).

The statistical tests applied to pseudorandom number generators for use in Monte Carlo simulations are generally simple; for cryptographic applications, pseudorandom number generators must pass a far more stringent battery of statistical tests.

Now we are ready to give a precise definition of a secure pseudorandom bit generator. To achieve insensitivity to the computational model, this definition, due to Yao (1982), is an asymptotic one. Instead of a single generator *G* of fixed size, we consider an ensemble $\mathcal{G} = \{G_k : k \geq 1\}$ of generators, with

$$G_k : \{0,1\}^k - \{0,1\}^{f(k)},$$

that have the property of being **polynomial length-increasing**; i.e.,

$$k + 1 \leq f(k) \leq k^m + m,$$

for some fixed integer $m \geq 1$. We say that $\mathcal{G}$ is a **nonuniform secure pseudorandom bit generator** (abbreviated N-PRG) if

1. There is a (deterministic) polynomial-time algorithm that computes *all* G_k(x), given (*k*, x) as input with $x \in \{0,1\}^k$, and
2. For any PSIZE family $\mathcal{F}$ of Boolean circuit statistical tests, G_k(U_k) is **polynomially indistinguishable** by $\mathcal{F}$ from *U*f(*k*); i.e., for all $m \geq 1$ the distribution $G_k(U_k)$ is k^{-m}-indistinguishable by $\mathcal{F}$ from *U*f(*k*) for all sufficiently large *k*.

Condition 1 says that $\mathcal{G}$ is easy to compute. Condition 2 requires more explanation. A **Boolean circuit** C_ℓ is made of *and, or*, and *not* gates, having ℓ inputs, and computes a Boolean function $\hat{C}_\ell : \{0,1\}^\ell \to \{0,1\}$. A family $\mathcal{F} = \{C_k : k \geq 1\}$ of Boolean circuits is of **polynomial size** (abbreviated $\mathcal{F} \in \mathrm{PSIZE}$) if there is some fixed *m* such that the number of gates in C_k is $\leq k^m + m$, for all *m*. The Boolean function $C_\ell(x)$ computed by C_ℓ with $\ell = f(k)$ provides a statistical test for comparing $G_k(U_k)$ and U_ℓ. Then condition 2 asserts that

$$\left| E[C_k(x) : x \in G_k(U_k)] - E[C_k(x) : x \in U_{f(k)}] \right| < k^{-m}, \text{ for all } k > c_0(\mathcal{F}, m).$$

The PSIZE measure of computational complexity is **nonuniform** in that no relation between the circuits C_k for different *k* is required. In particular, the family $\mathcal{F}$ might be noncomputable (nonrecursive). Condition 2 may be colloquially rephrased as, "$\mathcal{G}$ passes all polynomial size statistical tests."

This definition implies that N-PRGs must have a **one-way property**: Given the output $y = G_k(x) \in \{0,1\}^{f(k)}$, one cannot recover the string $x \in \{0,1\}^k$ using a PSIZE family of circuits for a fraction of the inputs exceeding 1/(k^m+*m*), for any fixed *m*. Otherwise, it would fail the statistical test, "Is there a seed x such that G_k(x) = y?"

We also consider the weaker notion of **uniform secure pseudorandom bit generator** (U-PRG) in which PSIZE in condition 2 is replaced by PTIME, where a family $\mathcal{F} = \{C_k\}$ of circuits is in PTIME if there is a polynomial-time algorithm for computing C_k, given 1^k as input. This condition may be rephrased as, "$\mathcal{G}$ passes all polynomial time statistical tests." Since $\mathbf{PTIME} \subseteq \mathbf{PSIZE}$, an N-PRG is always a U-PRG.

These definitions of N-PRG and U-PRG form the basis of a nice theory, whose purpose is to reduce the existence question for apparently very complicated objects (N-PRGs) to the existence question for simpler, more plausible objects. As an example, we have

Theorem 6.1 *If there exists an N-PRG with* $f(x) = k + 1$, *then there exists an N-PRG with* $f(k) = x^m + m$ *for each* $m \geq 2$. *(Similarly for U-PRGs.)*

That is, if there is an PRG that can inflate k random bits to $k + 1$ pseudorandom bits, this is already sufficient to construct a PRG that inflates k random bits to any polynomial number of pseudorandom bits. A proof appears in Boppana and Hirschfeld (1989).

One disadvantage of these definitions (N-PRG, etc.) at present is that *no such generators have been proved to exist*. In fact, proving that they exist is at least as hard a problem as settling the famous $P \neq NP$ question of theoretical computer science. On the positive side, however, theorem 6.4 below suggests that the RSA generator may well be a U-PRG.

Yao (1982) provided the basic insight that the nature of secure pseudo-randomness (N-PRG) is the *computational unpredictability* of successive bits produced by a pseudorandom bit generator. The notion of computational unpredictability is captured by the concept of a **nonuniform next-bit test**, as follows. Let $\mathcal{G} = \{G_k\}$ be a collection of mappings $G_k : \{0,1\}^k \rightarrow \{0,1\}^{f(k)}$, where $f(k)$ is polynomial in k. A **predicting collection** is a doubly-indexed family of Boolean circuits $\mathcal{C} = \{C_{i,k} : 1 \leq i \leq f(k) - 1,\ k \geq 1\}$ computing functions $C_{i,k} : \{0,1\}^i \rightarrow \{0,1\}$, where the circuit $C_{i,k}$ is viewed as predicting the $(i + 1)$st bit of a sequence in $\{0,1\}_{f(k)}$ given the first i bits. The predicting collection is **polynomial size** if there is some $m \geq 2$ such that each $C_{i,k}$ has at most $k^m + m$ gates. Let $p^{\mathcal{C}}_{i,k}$ denote the probability that the output of $C_{i,k}$ applied to a bit sequence $(x_1, \ldots ,x_i)$ drawn from $G_k(U_k)$ is x_{i+1}. We say that $\mathcal{G}$ *passes the nonuniform next-bit test* if for each polynomial-size predicting collection $\mathcal{C}$ and each $m \geq 2$, for all sufficiently large k,

$$\left| p^{\mathcal{C}}_{i,k} - \frac{1}{2} \right| < \frac{1}{k^m + m}$$

holds for $1 \leq i \leq f(k)-1$. This test says that x_{i+1} cannot be accurately guessed using PSIZE circuit families with $x_1, ..., x_i$ as input.

Theorem 6.2 (Yao, 1982) *The following are equivalent*:

1. *A collection $\mathcal{G} = \{G_k\}$ passes the nonuniform next-bit test.*
2. *A collection $\mathcal{G} = \{G_k\}$ passes all polynomial-size statistical tests.*

In this theorem, there is no restriction on the sizes of circuits needed to compute the functions G_k; e.g., they could be of exponential size in k. The direction $(2) \Longrightarrow (1)$ is fairly easy to prove because the next-bit test is essentially a family of PSIZE statistical tests. The direction $(1) \Longrightarrow (2)$ is harder. A proof of this result is given in Boppana and Hirschfeld (1989).

Corollary 1 *A polynomial-time computable collection $\mathcal{G} = \{G_k\}$ that passes the nonuniform next-bit test is an N-PRG.*

Pseudorandom bit generators must have a very strong one-way property: from the output of such a generator, it must be computationally infeasible to extract any information at all about its input, for essentially all inputs. **One-way functions** are required only to satisfy a weaker one-way property: they are easy to compute but difficult to invert on a nonnegligible fraction of their instances.

A formal definition of one-way function is as follows: A **nonuniform one-way collection** $\{f_k\}$ consists of a family of functions $f_k : \{0,1\}^k \rightarrow \{0,1\}^{p(k)}$, where $k \leq p(k) \leq k^m + m$ for some fixed m, such that

1. The ensemble $\{f_k\}$ is uniformly polynomial-time computable, and
2. For each PSIZE circuit family $\{C_k\}$ with $C_k : \{0,1\}^{p(k)} \rightarrow \{0,1\}^k$, the functions $\{f_k\}$ are hard to invert on a nonnegligible fraction $1/(k^c + c)$ of their inputs, where c is a fixed positive constant depending on $\{C_k\}$. That is, for x drawn from the uniform distribution on $\{0, 1\}^k$,

$$P[f_k(x) = f_k(C_k(f_k(x)))]$$

is at most $1 - (k^c + c)^{-1}$ for all large enough k (depending on $\{C_k\}$).

The one-way property embodied in this definition is weaker than that required of an N-PRG in several ways: a function f_k can have an extremely nonuniform probability distribution on its range $\{0, 1\}$ $p(k)$, it is required to be hard to invert only on a possibly small fraction $1/(k^c + c)$ of its inputs, and even on those inputs where it is hard to invert, some partial information about its inverse may be easy to obtain.

Impagliazzo et al. (1989) showed that one can bootstrap a nonuniform one-way collection into an N-PRG:

Theorem 6.3 *The following are equivalent*:

1. *There exists a nonuniform polynomial pseudorandom bit generator* (*N-PRG*).
2. *There exists a nonuniform one-way collection* $\{f_k\}$ *of functions*.

The implication $(1) \Longrightarrow (2)$ is easy, because an N-PRG $\mathcal{G} = \{G_k\}$ is a one-way collection. To prove $(2) \Longrightarrow (1)$, it suffices by theorem 6.1 to enlarge a set of ℓ random bits to obtain $\ell + 1$ pseudorandom bits, for an infinite set of ℓ. We may reduce to the case that the function f_k is length-preserving by padding its input bits if necessary. The first key idea, due to Goldreich and Levin (1989), is that the Boolean inner product

$$B(\mathbf{x}, \mathbf{r}) = \sum_{i=1}^{k} x_i r_i \pmod 2$$

is a **hard-core predicate** for all length-preserving one-way families $f : \{0,1\}^k \rightarrow \{0,1\}^k$. That is, the probability distribution of the $(2k+1)$-bit strings (f(x), **r**, B(x, **r**)) for (x, **r**) drawn uniformly from $\{0, 1\}^{2k}$ is polynomially indistinguishable from (f(x), **r** , β), where β is a truly random bit drawn independent of x and **r**. We are now done if f is a permutation, because then (f(x), **r**, β) would be uniformly distributed on $2k + 1$ bits. The difficult case occurs when f is not one-to-one, in which case the set $S_x = \{y: f(y) = f(x)\}$ may be large. In this case, f(x) contains only about k-$\log_2 |S_x|$ bits of information, and so the entropy of (f(x), **r**, β) may be smaller than $2k$, and we gain no computational entropy. The key idea of Impagliazzo et al. (1989) was to add in $\log_2 |S_x|$ extra bits of information from x by hashing x with a random hash function $h : \{0,1\}^k \rightarrow \{0,1\}^{|S_x|}$. If one can do this, then the distribution of (f(x), **r**, h, h(x), B(x, **r**)) will be polynomially indistinguishable from (f(x), **r**, h, h(x), β) and one extra pseudorandom bit will have been created. The nonuniform model of computation is used in determining the number $\log_2 |S_x|$ of bits to hash in the hash function for x.

Hastad (1990) proved that a uniform version of theorem 6.3 holds; i.e., one may replace "nonuniform" by "uniform" in (1) and (2).

Finally, secure PRGs can be used to construct secure private-key cryptosystems and vice versa. A **private-key cryptosystem** uses a (private) key exchanged by some secure means between two users, the possession of which enables them both to encrypt and decrypt messages sent between them. Encrypted messages should be unreadable to anyone else. They should appear random to any unauthorized receiver, and ideally no statistical information at all should be extractable from the encrypted message. This is seldom achieved in actual cryptosystems, and statistical methods are one of the staples of cryptanalysis. Absolute security can always be achieved by using a key of the same length as the totality of encrypted messages to be exchanged (the "**one time pad**"). How much security is possible when the key is to be shorter than the messages to be encrypted? An analogy between pseudorandom number generation and private-key cryptosystems is apparent: the key supplies absolutely random bits from which pseudorandom bits (the encrypted message) are created. This analogy can be made precise. For a statement of the result, see Lagarias (1990, §6); some related results appear in Impagliazzo and Luby (1989).

Rompel (1990) showed how to construct a secure authentication scheme based on a one-way function.

Finally, we mention that probabilistic algorithms play a fundamental role in the theoretical foundations of modern cryptography, as indicated in Goldwasser and Micali (1984).

6.4 PERFORMANCE OF CERTAIN PSEUDORANDOM BIT GENERATORS

We now turn to results on the performance of various generators; see Knuth (1981) for general background.

Linear congruential pseudorandom number generators have been extensively studied, and many of their statistical properties carefully analyzed. With proper choice of parameters, they can produce sequences $\{x_k\}$ such that $\{x_k \pmod{p} : 1 \le k \le M\}$ is close to uniformly distributed on [1, M]. Marsaglia (1968) was the first to notice that these generators have certain undesirable correlation properties among several consecutive iterates (x_n, x_{n+1}, ..., *x*n+*k*). Extensive analysis is given in Niederreiter (1978). These generators are apparently unsuitable for cryptographic uses (see Frieze et al., 1988).

The group-theoretic structure of certain other generators can be used to prove "almost-everywhere hardness" results for such generators, including the following one:

RSA Bit Generator. *Given* $k \geq 2$ *and* $m \geq 1$, *select odd primes* p_1 *and* p_2 *uniformly from the range* $2^k \leq p_i < 2^{k+1}$ *and form* $N = p_1p_2$. *Select e uniformly from* $[1, N]$ *subject to* $(e, \quad (N)) = 1$. *Set*

$$x_{n+1} \equiv (x_n)^e \pmod{N},$$

and let the bit z_{n+1} *be given by*

$$z_{n+1} \equiv x_{n+1} \pmod{2}.$$

Then $\{z_n : 1 \leq n \leq k^m + m\}$ *are the pseudorandom bits generated from the seed* x_0 *of length* $2k$ *bits*.

We consider the problem of *predicting* the next bit z_{n+1} of the RSA generator, given $\{z_1, ..., z_n\}$. Alexi et al. (1988) showed that getting any information at all about z_{n+1} is as hard as the (apparently difficult) problem of factoring *N*. Their argument shows how an oracle that guesses the next bit with a probability slightly greater than 1/2 can be used in an explicit fashion to construct an inversion algorithm.

Theorem 6.4 *Let* (e, N) *be a fixed pair, where* $(e, \quad (N)) = 1$ *and* $N = p_1p_2$ *is odd, with the associated power generator*

$$E(y) \equiv y^e := x \pmod{N}.$$

Suppose one is given a 0-1 valued oracle function $O(x)$ *for which*

$$O(x) \equiv y \pmod{2}$$

holds for $\left(\frac{1}{2} + \frac{1}{k^m+m}\right)$ *N of all inputs* $x \in [0, N-1]$, *where* $k = \log_2 N$ *Then there is a probabilistic polynomial time algorithm that makes* $O(k^{C_0 m})$ *oracle calls, uses* $O(k^{C_0 m})$ *"random" bits, halts in* $O(k^{C_0 m})$ *steps, and for any x finds y with probability* $\geq 1 - \frac{1}{N}$, *where the probability is taken over all sequences of "random" bits*.

This algorithm involves several clever ideas. The first of these is due to Ben-Or et al. (1983), using the fact that the encryption function $E(\cdot)$ respects the group law on $(\mathbf{Z}/N\mathbf{Z})^*$; i.e.,

$$E(ay) = E(a)E(y).$$

They suppose that one is given an oracle O_I that perfectly recognizes membership in an interval $I = \left[-\frac{1}{2}\delta N, \frac{1}{2}\delta N\right]$ for fixed positive $\delta \leq \frac{1}{2}$; i.e.,

$$O_I(x) = \begin{cases} 1 & \text{if } y \in I \\ 0 & \text{if } y \notin I. \end{cases}$$

Define $[x]_N$ to be the least nonnegative residue (mod N) and define $abs_N(x)$ to be the distance of the least (positive or negative) residue from 0 so that

$$abs_N(x) = \begin{cases} [x]_N & 0 \leq x < N/2 \\ N - [x]_N & N/2 < x < N. \end{cases}$$

Let $par_N(x)$ be the parity of $abs_N(x)$. The clever idea is to generate "random" a, b and attempt to compute the *gcd* of $[ay]_N$ and $[by]_N$ using a modified binary *gcd* algorithm due to Brent and Kung (1983) and Purdy (1983). This algorithm computes the greatest common divisor (r, s) by noting that it is equal to $2\left(\frac{r}{2}, \frac{s}{2}\right)$ if r, s both even, to $(\frac{r}{2}\ s)$ or $(r, \frac{s}{2})$ if one is even, and to $\left(\frac{r+s}{2}, \frac{r-s}{2}\right)$ otherwise. It is easy to check that $\max(|r|, |s|)$ is nonincreasing at each iteration and that it decreases by a factor of 3/4 over every two iterations, so this algorithm halts in at most $2\log_{4/3} N$ iterations. The Ben-Or et al. algorithm draws pairs (a, b) of "random" numbers uniformly on $[0, N]$. Such a pair will be called "good" (for the unknown y) if $[ay]_N$ and $[by]_N$ both fall in I. For a good pair the oracle O_I can be used to determine correctly the parity of $[ay]_N$ and $[by]_N$, even though y is unknown. We use the fact that if $w \in I$ is even then $\frac{w}{2} \in I$, whereas if it is odd then $\frac{w}{2} \in \left[\frac{N}{2} - \frac{|I|}{2}, \frac{N}{2} + \frac{|I|}{2}\right]$. Hence for a good pair, one has $\frac{1}{2}y \in I$ if $[ay]_N$ is even and $\frac{1}{2}y \notin I$ if $[ay]_N$ is odd. This holds similarly for $[by]_N$. Thus using the group law

$$\begin{aligned} par_N([ay]_N) &= 1 - O_I\left(E\left[\left(\frac{a}{2}\right)y\right]\right) \\ &= 1 - O_I\left(E\left[\frac{a}{2}\right]x\right) \end{aligned}$$

is calculable. Consequently, one can compute one step of the modified binary *gcd* algorithm to ($[ay]_N$, $[by]_N$) and obtain new values ($[a'y\,]_N$, $[b'y]_N$), with (a', b') known. Now (a', b') is still a good pair, so we can continue to follow the modified binary *gcd* algorithm perfectly and eventually determine $[ly]_N$ = *gcd* ($[ay]_N$, $[by]_N$), where l is known. We say that the algorithm succeeds if $[ly]_N = 1$, because in that case $y \equiv l^{-1}$ (mod N) is found. We verify success in this case by checking that $E[l^{-1}] = x$. In all other cases, when (a, b) is not good or when it is good and the *gcd* is not 1, we carry out the above procedure for $2\log_{4/3} N$ iterations and check the final iterate l to see whether $E[l^{-1}] = x$.

This procedure is a probabilistic polynomial-time algorithm. The pair (a, b) is good with probability at least σ^2. Also, because the distributions of $[ay]_N$, $[by]_N$ are uniform on $\left[-\frac{1}{2}\delta N, \frac{1}{2}\delta N\right]$ and because the probability that *gcd* (r, s) = 1 for such draws approaches [illegible] , it exceeds a positive constant α for all values of $\{|l|\}$. Thus the success probability for any draw (a, b) is $\geq \delta^2\alpha$, and the expected time until the algorithm succeeds is polynomial in log N.

One immediately sees that this algorithm still works if the oracle O_{I} is not perfect but is accurate with probability of error less than $(4\log_{4/3}N)^{-1}$.

Subsequent work of several authors showed how an oracle having a 1/2 + 1/(log $N)^k$ advantage in guessing parity of the smallest RSA bit can be used to simulate a much more accurate oracle and then to simulate an oracle such as O_{I} above with sufficient accuracy to obtain theorem 6.4 (see Alexi et al., 1988).

Theorem 6.4 shows that if the factoring problem is difficult, then recovering any information at all about z_{n+1} given $\{z_1, ..., z_n\}$ is hard. In particular, the bits $\{z_k\}$ would then be computationally indistinguishable from i.i.d. coin flips.

One can extract several pseudorandom bits from each of the iterates x_n of the RSA generator. The strongest result to date on this problem can be found in Schrift and Shamir (1990).

The use of the group structure on (**Z**/*N***Z**)* to "randomize" is an example of the concept of an **instance-hiding scheme** discussed in Chapter 5.

Acknowledgment. This chapter is an abridged and revised version of Lagarias (1990), published by the American Mathematical Society.

REFERENCES

Alexi, W., B. Chor, O. Goldreich, and C.P. Schnorr (1988), RSA and Rabin functions: Certain parts are as hard as the whole, *SIAM J. Comput*. **17**, 194-209.

Ben-Or, M., B. Chor, and A. Shamir (1983), On the cryptographic security of single RSA bits, in *Proc. 15th Annual Symposium on Theory of Computing* , ACM Press, New York, 421-430.

Blum, L., M. Blum, and M. Shub (1986), A simple unpredictable pseudorandom number operator, *SIAM J. Comput*. **15**, 364-383. (Preliminary version: CRYPTO '82.)

Blum, M., and S. Micali (1982), How to generate cryptographically strong versions of pseudorandom bits, in *Proc. 22nd Annual Symposium on Foundations of Computer Science*, IEEE Computer Society Press, Los Alamitos, Calif., 112-117. (Final version: *SIAM J. Comput*. **13** (1984), 850-864.)

Boppana, R., and R. Hirschfeld (1989), Pseudorandom generators and complexity classes, in *Randomness and Computation*, S. Micali, ed., Advances in Computing Research, vol. 5, JAI Press, Greenwich, Conn., 1-26.

Brent, R.P., and H.T. Kung (1983), Systolic VLSI arrays for linear time *gcd* computations, in *VLSI 83, IFIP*, F. Anceau and D.J. Aas, eds., Elsevier Science Publishers, New York, 145-154.

Brudno, A.A. (1982), Entropy and the complexity of the trajectories of a dynamical system, *Trans. Moscow Math. Soc*. **44**, 127-151.

Chaitin, G.J. (1966), On the length of programs for computing finite binary sequences, *J. Assoc. Comput. Mach*. **13**, 547-569.

Collet, P., and J.-P. Eckmann (1980), *Iterated Maps on the Interval as Dynamical Systems*, Birkhäuser, Boston.

Devroye, L. (1986), *Nonuniform Random Variate Generation*, Springer-Verlag, New York.

Diffie, W., and M. Hellman (1976), New directions in cryptography, *IEEE Trans. Inform. Theory* **22**, 644-654.

Frieze, A, J. Hastad *, R. Kannan, J.C. Lagarias, and A. Shamir (1988), Reconstructing truncated integer variables satisfying linear congruences, *SIAM J. Comput*. **17**, 262-280.

Garey, M., and D. Johnson (1979), *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, San Francisco.

Goldreich, O., and L. Levin (1989), A hard-core predicate for all one-way functions, in *Proc. 21st Annual Symposium on Theory of Computing* , ACM Press, New York, 25-32.

Goldreich, O, H. Krawczyk, and M. Luby (1988), On the existence of pseudorandom generators, in *Proc. 29th Annual Symposium on Foundations of Computer Science*, IEEE Computer Society Press, Los Alamitos, Calif., 12-24.

Goldwasser, S, and S. Micali (1984), Probabilistic encryption, *J. Comput. Syst. Sci*. **28**, 270-299. (Preliminary version: *Proc. 14th Annual Symposium on Theory of Computing* (1982), ACM Press, New York, 364-377.)

Hastad*, J. (1990), Pseudo-random generators under uniform assumptions, in *Proc. 22nd Annual Symposium on Theory of Computing*, ACM Press, New York, 395-404.

Impagliazzo, R., and M. Luby (1989), One-way functions are essential for complexity-based cryptography, in *Proc. 30th Annual Symposium on Foundations of Computer Science*, IEEE Computer Society Press, Los Alamitos, Calif., 230-235.

Impagliazzo, R., L. Levin, and M. Luby (1989), Pseudorandom number generation from one-way functions, in *Proc. 21st Annual Symposium on Theory of Computing*, ACM Press, New York, 12-24.

Karp, R., and R. Lipton (1982), Turing machines that take advice, *L'En-seignement Math*. **28**, 191-209. (Preliminary version: Some connections between nonuniform and uniform complexity classes, in *Proc. 12th Annual*

Symposium on Theory of Computing (1980), ACM Press, New York, 302-309.)

Knuth, D. (1981), *The Art of Computer Programming, vol. 2: Seminumerical Algorithms* (second ed.), Addison-Wesley, Reading, Mass.

Knuth, D., and A.C. Yao (1976), The complexity of nonuniform random number generation, in *Algorithms and Complexity*, J.F. Traub, ed., Academic Press, New York, 357-428.

Kolmogorov, A.N. (1965), Three approaches to the concept of "the amount of information," *Probl. Inform. Transm*. 1, 3-13.

Lagarias, J.C. (1990), Pseudorandom number generators in number theory and cryptography, in *Cryptology and Computational Number Theory*, C. Pomerance, ed., Proc. Syrup. Appl. Math. No. 42, American Mathematical Society, Providence, R.I., 115-143.

Marsaglia, G. (1968), Random numbers fall mainly in the planes, *Proc. Nat. Acad. Sci. USA* **61**, 25-28.

Marsaglia, G. (1985), A current view of random number generators, in *Computer Science and Statistics: Sixteenth Symposium on the Interface, L.* Billard, ed., North-Holland, New York, 3-11.

Marsaglia, G., and A. Zaman (1991), A new class of random number generators, *Ann. Appl. Probab*. 1, 462-480.

Martin-Lof, P. (1966), The definition of random sequences, *Inform. Contr*. **9**, 619-682.

Meyer, A.R., and E.M. McCreight (1971), Computationally complex and pseudorandom zero-one valued functions, in *Theory of Machines and Computations*, Z. Kohlavi and A. Paz, eds., Academic Press, New York, 19-42.

Niederreiter, H. (1978), Quasi-Monte Carlo methods and pseudorandom numbers, *Bull. Amer. Math. Soc*. 84, 957-1041.

Odlyzko, A.M. (1985), Discrete logarithms in finite fields and their cryp tographic significance, in *Advances in Cryptology: Eurocrypt '84*, Lecture Notes in Computer Science no. 209, Springer-Verlag, New York, 224-316.

Purdy, G.B. (1983), A carry-free algorithm for finding the greatest common divisor of two integers, *Comput. Math. Appl.* **9**, 311-316.

Reeds, J. (1979), Cracking a multiplicative congruential encryption algorithm, in *Information Linkage Between Applied Mathematics and Industry*, P.C. Wong, ed., Academic Press, New York, 467-472.

Rivest, R., A. Shamir, and L. Adleman (1978), On digital signatures and public key cryptosystems, *Commun. ACM* **21**, 120-126.

Rompel, J. (1990), One-way functions are necessary and sufficient for secure signatures, in *Proc. 22nd Annual Symposium on Theory of Computing*, ACM Press, New York, 387-394.

Rubinstein, R.Y. (1982), *Simulation and the Monte Carlo Method*, John Wiley & Sons, New York.

Savage, J. (1976), *The Complexity of Computing*, John Wiley & Sons, New York.

Schrift, A., and A. Shamir (1990), The discrete log is very discreet, in *Proc. 22nd Annual Symposium on Theory of Computing*, ACM Press, New York, 405-415.

Schuster, H.G. (1988), *Deterministic Chaos*, VCH Verlagsgesellschaft mbH, Weinheim, Germany.

Wolfram, S. (1986), Random sequence generation by cellular automata, *Adv. Appl. Math.* **7**, 123-169.

Yao, A. (1982), Theory and applications of trap door functions (extended abstract), in *Proc. 23rd Annual Symposium on Foundations of Computer Science*, IEEE Computer Society Press, Los Alamitos, Calif., 80-91.

Yao, A. (1988), Computational information theory, in *Complexity in Information Theory*, Y. Abu-Mostafa, ed., Springer-Verlag, New York, 1-15.

7. Probabilistic Analysis of Packing and Related Partitioning Problems

E.G. Coffman, Jr., D.S. Johnson, P.W. Shor
AT&T Bell Laboratories
and
G.S. Lueker
University of California at Irvine

ABSTRACT

In the last 10 years, there have been major advances in the average-case analysis of bin packing, scheduling, and similar partitioning problems in one and two dimensions. These problems are drawn from important applications throughout industry, often under the name of stock cutting. This chapter briefly surveys many of the basic results, as well as the probabilistic methods used to obtain them. The impact of the research discussed here has been twofold. First, analysis has shown that heuristic solutions often perform extremely well on average, and hence can be recommended in practice, even though worst-case behavior can be quite poor. Second, the techniques of applied probability that have developed for the analysis of bin packing have found application in completely different arenas, e.g., statistics and stochastic models.

7.1 INTRODUCTION

7.1.1 Problems

The problems studied here all involve the partitioning of a set of positive numbers into a collection of subsets satisfying a sum constraint. The following two problems are among the most fundamental. They have wide-ranging applications throughout computer science and operations research (see, e.g., Coffman et al., 1984b; Coffman et al., 1988; and Dyckhoff, 1990).

Bin Packing (BP). *Given* $c > 0$ *and a set* $S = \{X_1, ..., X_n\}$ *with* $0 < X_i \leq c$, $1 \leq i \leq n$, *partition S into a minimum number of subsets such that the sum of the* X_i's in each subset is no more than c.

The X_i's are usually called items or pieces and are thought of as being packed into bins B_1, B_2, ..., each with capacity *c*; the items packed in a bin constitute one of the subsets in a solution to the optimization problem.

Multiprocessor Scheduling (MS). *Given an integer* $m \geq 1$ *and a set* $S = \{X_1, ..., X_n\}$, *partition S into m subsets such that among all such partitions, the maximum subset sum is minimized.*

Note that the MS problem is complementary to the BP problem in that the objective function and the given parameter are interchanged. The items are now called tasks or jobs, with running times or durations instead of sizes. The bins become processors P_1, ..., P_m, and the partition becomes a schedule of *S* on *m* processors that minimizes the **makespan** *c*, i.e., the completion time of a latest finishing task. Because of the sequential nature of most heuristics, it is convenient to assume that the set to be partitioned is given as a list $L_n = (X_1, ..., X_n)$ from which items are packed or scheduled one by one. If *H* denotes an MS heuristic, then $H(L_n, m)$ denotes the makespan of the *m*-processor schedule generated by *H* for the tasks in L_n. In the BP problem the bin capacity is only a scale factor, so we take $c = 1$ without loss of generality. Thus, if *H* denotes a BP heuristic, then $H(L_n)$ denotes the number of unit capacity bins in which *H* packs the items of L_n.

Merely deciding whether a list of numbers can be partitioned into two subsets with equal sums is NP-complete, so, as one would expect, both the BP and the MS problems are NP-complete. Thus, one is unlikely to find an algorithm that will solve these problems exactly and efficiently.[1] For this reason, a large literature has built up over the past 20 years on the design and analysis of heuristic or approximation algorithms. Such algorithms are designed to generate optimal or nearly optimal solutions for most problem instances. Quantifying this last statement is the goal of analysis.

7.1.2 Analysis

Early research on BP, MS, and related problems concentrated on combinatorial, worst-case results, as reflected in the survey by Coffman et al. (1984b). For example, a scheduling heuristic H would be assessed by determining for each m a least upper bound over all L_n, and n on the ratio $H(L_n, \mathrm{m})/\mathrm{OPT}(L_n, \mathrm{m})$, where OPT stands for an optimal algorithm; i.e., $\mathrm{OPT}(L_n, \mathrm{m})$ denotes the makespan of a solution to the MS problem for the problem instance (L_n, m). Similarly, the ratios $H(L_n)/\mathrm{OPT}(L_n)$ were bounded for BP heuristics H. Such results are inherently pessimistic, so probability models were introduced in order to learn more about the probable or average-case behavior of heuristics.

Probabilistic analysis began about 10 years ago and gained considerable momentum when some striking new results were developed a few years later. In the standard probability model, the X_i's are taken as independent samples of a random variable X with a given distribution $F(x)$. The goal is then an estimate of distributions such as $P[H(L_n) \leq x]$, or what is sometimes easier to obtain, expected values such as $E[H(L_n, \mathrm{m})]$, where the expectations are over all n-item samples $L_n = (X_1, ..., X_n)$.

Typically, exact analysis of probability models is quite difficult, especially for the more efficient algorithms, so asymptotic techniques have been used. These techniques estimate behavior for large problem instances, i.e., for large n. Also, the estimates often take the form of expressions with terms that are precise only within unspecified multiplicative constants. For example, let $F(x)$ be the uniform distribution on [0,1]. Then as illustrated later, there are BP heuristics H for which it has been proved that $E[H(L_n)] = n/2 + \Theta(\sqrt{n})$. Here, the $(\cdot)$ notation is simply a relaxation of the concept "is proportional to." Precisely, $f(n) = \ (g(n))$ means that there exist constants $\alpha, \beta > 0$ such that for all n large enough,

$$\alpha g(n) \leq f(n) \leq \beta g(n).$$

[1] For a comprehensive treatment of NP-completeness and its implications, see Garey and Johnson (1983).

If we only know the existence of $\beta > 0$ such that the right-hand inequality is satisfied for all n large enough, then we write the familiar $f(n) = O(g(n))$. A similar restriction to α and the left-hand inequality is denoted $f(n) = \Omega(g(n))$.

We emphasize that usually very little is known about the multiplicative constants hidden in the $(\cdot)$ terms. One can almost always find some bounds for these constants, but in most cases there is reason to believe that the bounds are very crude.

In the remainder of this section, we present a number of fundamental algorithms together with a sampling of results that measure the quality of the packings or schedules produced.

7.1.3 Bp Algorithms

We begin by describing three algorithms that pack the items in the sequence $X_1, ..., X_n$. An item is packed when it is encountered; once packed, it is not moved thereafter. The algorithms are said to be **on-line** because, for each i, $1 \leq i \leq n$, the rule that decides where X_i is packed is independent of the number and sizes of the remaining items $X_{i+1}, ..., X_n$. All three algorithms begin by packing X_1 into B_1.

The simplest of the three algorithms is **next fit (NF)**. In packing X_i, $i \geq 2$, NF first checks the highest-indexed, nonempty bin, say B_j, $j \geq 1$. X_i is packed in B_j if it fits, i.e., if X_i plus the sum of the items already packed in B_j is at most 1. Otherwise, X_i is packed into B_{j+1}, which then becomes the new highest-indexed, nonempty bin.

The two algorithms, **first fit (FF)** and **best fit (BF)**, improve on NF by checking *all* nonempty bins before starting a new bin; i.e., FF and BF pack am item X_i into an empty bin if and only if X_i does not fit into any nonempty bin. FF packs X_i, $i \geq 2$, into the lowest-indexed nonempty bin, if any, in which X_i fits, and BF packs X_i into a nonempty bin, if any, in which X_i fits best, i.e., with the least unused capacity left over. Ties are resolved by BF in favor of lower-indexed bins.

Improved, off-line versions of these algorithms are obtained by first sorting the X_i's into decreasing order; the corresponding **NFD, FFD** , and **BFD** algorithms (D stands for decreasing) are simply NF, FF, and BF applied to the list $(X(n)), ..., X_{(1)})$, where $X((i))$ denotes the ith smallest item in L_n.

Table 7.1 summarizes a number of the basic results that have been derived for the above algorithms under the assumption $X \sim U(0, 1)$; i.e., $F(x)$ is the uniform distribution on [0, 1]. The performance metric shown is the

TABLE 7.1: Analyses of Bin Packing Algorithms

Assumption: $X \sim$ U(0, 1)
Next Fit: $E[W^{NF}(L_n)] \sim \frac{n}{6}$ Coffman et al. (1980)
First Fit: $E[W^{FF}(L_n)] = \quad (n^{2/3})$ Shor (1986) and Coffman et al. (1991)
Best Fit: $E[W^{BF}(L_n)] = \Theta(\sqrt{n}\log^{3/4} n)$ Shor (1986)
Next Fit Decreasing: $E[W^{NFD}(L_n)] = (.145 \ldots)n$ Hofri and Kamhi (1986) and Csirik et al. (1986)
First Fit Decreasing (FFD), Best Fit Decreasing (BFD), and Optimal (OPT): $E[W^H(L_n)] = \Theta(\sqrt{n})$ for H = FFD (Bentley et al, 1984) BFD, or OPT (Lueker, 1982)

expected wasted space,

$$
\begin{aligned}
E[W^H(L_n)] &= E[H(L_n) - \sigma(L_n)] \qquad (7.1)\\
&= E[H(L_n)] - \frac{n}{2},
\end{aligned}
$$

where $\sigma(L_n)$ denotes the sum of the item sizes in L_n. Note that FF and BF and their counterparts FFD and BFD are all asymptotically optimal in the sense that the ratio of expected wasted space $E[W_n^H]$ to the expected occupied space $E[\sigma(L_n)] = n/2$ tends to 0 as $n \rightarrow \infty$. This is in sharp contrast to worst-case bounds, which show that, for infinitely many $n > 0$,

$W^H(L_n)/\sigma(L_n)$ can be as large as 7/10 for H = FF or BF and as large as 2/9 for H = FFD or BFD (Johnson et al., 1974).

The shortcomings of the (·) results are apparent, because the average-case results do not distinguish between FFD, BFD, and OPT. On the other hand, the average-case results do show that for all n sufficiently large, $E[FF(L_n)] > E[BF(L_n)]$; a comparable distinction does not appear in the worst-case results.

7.1.4 Ms Algorithms

We describe three algorithms. The simplest is the on-line **list scheduling** (LS) algorithm, which schedules the tasks in the given sequence $X_1, ..., X_n$ on the processors $P_1, ..., P_m$, with X_1 starting on P_1. LS schedules X_i, $i \geq 2$, on that processor having the smallest workload in the schedule for $X_1, ..., X_{i-1}$, with ties broken in favor of lower-indexed processors. By the workload of a processor, we mean the total duration of the tasks already scheduled on that processor. As before, LS can be improved by first sorting L_n into decreasing order. LS along with the initial sorting is called the **largest processing time (LPT)** algorithm.

The third MS heuristic was originally proposed for a somewhat different optimization problem: With the instances (L_n, m) the same as in the MS problem, the objective of the **set-partitioning (SP)** problem is to find a schedule that minimizes the difference in the maximum and minimum processor workloads. Clearly, one expects a good heuristic for SP to be a good heuristic for MS; indeed, the two problems are obviously identical for $m = 2$. The heuristic described below is a **set-differencing method** (Karmarkar and Karp, 1982). It can be extended to all $m \geq 2$, but we confine ourselves to the case $m = 2$, because it is easier to describe and analyze.

Two tasks X and Y in list L are said to be **differenced in *L*** when a new list L' is formed from L by replacing X and Y with a task having duration $|X-Y|$. The **largest-first differencing (LFD)** heuristic applied to $L_n = L_n^{(1)}$ for $m = 2$ begins by differencing the largest two tasks in $L_n^{(1)}$ to form $L_n^{(2)}$. Then the largest two tasks are differenced in $L_n^{(2)}$ to form $L_n^{(3)}$. This procedure continues until a list $L_n^{(n)}$ of one task remains. LFD defines a schedule for L_n by requiring that the tasks differenced in $L_n^{(i)}$, $0 \leq i \leq n-1$, be scheduled on different processors and by requiring that the final processor workloads differ by the duration of the task in $L_n^{(n)}$. This schedule is easily developed by working backward through the sequence of differencing operations. First, the task in $L_n^{(n)}$ is put on one or the other of the two processors. Suppose the schedule for $L_n^{(i)}$, $i \geq 2$ has been formed, and let *X* and *Y* be the tasks differenced in $L_n^{(i-1)}$. Then the schedule for $L_n^{(i-1)}$ is formed from the schedule for $L_n^{(i)}$ by removing a task of duration |*X*-*Y*| and then scheduling *X* and *Y* on different processors so as to preserve the processor workload difference, i.e., the duration of the task in $L_n^{(n)}$.

In analogy with (7.1), typical illustrations of probabilistic results can be found in the analysis of the processor idle time averaged over the *m* processors,

$$A_{n,m}^{H} = \frac{mH(L_n, m) - \sigma(L_n)}{m}. \qquad (7.2)$$

We assume $m = 2$, so that $A^{H}(L_n, 2)$ is simply half the difference between the two processor finishing times in the schedule produced by *H*. A satisfactory analysis of LFD remains an open problem, but Karmarkar and Karp (1982) have studied a randomized, more easily analyzed modification of LFD that we denote LFD*. Results for LS, LPT, and LFD* with $X \sim U(0, 1)$ are shown in Table 7.2. In the order given, the algorithms are increasingly more complicated and increasingly more difficult to analyze, but they yield schedules that are increasingly more efficient for large *n*.

7.2 ANALYTICAL TECHNIQUES

We describe and illustrate below a number of the more important techniques that have been successfully applied to the analysis of BP and MS problems. A more extensive discussion appears in Coffman et al. (1988).

7.2.1 Markov Chains

For the simpler BP and MS heuristics, it is sometimes possible to formulate a tractable Markov chain that represents the element-by-element development of partitions. A state of the Markov chain must represent block sums in a suitable way; given the state space, the transition function is defined by the heuristic. Results for general *n* are obtained by a transient analysis, whereas asymptotics for large *n* are obtained by a steady state analysis.

To illustrate ideas, consider the average-case analysis of LS on $m = 2$ processors, and assume that $F(x)$ is the uniform distribution on [0, 1]. Define V_i as the (positive) difference between the processor finishing times after the first i tasks have been scheduled. The following recurrence is easily verified:

TABLE 7.2: Analysis of Makespan Scheduling Algorithms

List Scheduling:

*E[*A^{LS}(L_n, 2)] = 1/6
Feller (1971)

Largest Processing Time:

$$E[A^{LPT}(L_n,2)] \leq \frac{c}{2(n+1)}$$

Coffman et al. (1984a)

Modified Largest Difference First:

*There exists a c > 0 such that, with a probability that ten*ds to 1 *as* $n \rightarrow \infty$,

$A^{LFD*}(L_n, 2) = O(n^{-c\log n})$
Karmarkar and Karp (1982)

$$V_i = \begin{cases} |V_{i-1} - X_i|, & 1 \leq i \leq n, \\ 0, & i = 0 \,. \end{cases}$$

Since the X_i are i.i.d. random variables, $\{V_i\}_{i\geq 0}$ is a Markov chain. A routine analysis shows that the density for V_i is given by $f_i(x) = 2(1-x)$, for all $i > 2$ (Feller, 1971). Then we obtain the result cited in °7.1, viz. $E[A^{LS}(L_n, 2)] = E[V_n]/2 = 1/6$. Since $\mathrm{OPT}(L_n,2) \geq \sigma(L_n)/2$ and LS(L_n, 2) = $[V_n + \sigma(L_n)]/2$, we also have the relative performance bound

$$\frac{E[LS(L_n,2)]}{E[\mathrm{OPT}(L_n,2)]} \leq 1 + \frac{E[V_n]}{E[\sigma(L_n)]} = 1 + \frac{2}{3n} \,.$$

As another example, $\{NF(L_n), l_n\}_{n\geq 1}$ is a bivariate Markov chain, where l_n is the level, i.e., sum of item sizes, in the last bin of an NF packing of L_n. An analysis of this chain for $X \sim U(0, 1)$ shows that $E[W^{NF}(L_n)] = \frac{n}{6} + 6$, $n \geq 2$ (Hofri, 1984), thus refining the asymptotic result cited in °7.1. Indeed, an explicit expression for the distribution of NF(L_n) has been derived in Hofri (1984).

Unfortunately, Markov chain approaches seem to be limited to the relatively simplistic, less-efficient heuristics; the state spaces of Markov chains for other heuristics like FF and BF simply become too large and unwieldy.

7.2.2 Bounds

The immediate advantage of bounds is that they lead to a tractable analysis. The obvious sacrifice is that they provide only partial information. However, this information is often sufficient to choose between alternative heuristics. For example, the results cited in °7.1 for FF, BF, FFD, and BFD were all obtained by bounding techniques, yet they show that for all n sufficiently large, we have $E[\mathrm{FF}(L_n)] > E[\mathrm{BF}(L_n)] > E[H(L_n)]$, where H stands for either FFD or BFD. As illustrated below, bounding techniques have appeared in two basic forms.

Bounding the Objective Function. In analyzing the BP heuristic H, it may be possible to find a function $g(L_n)$ such that $g(L_n) \geq H(L_n)$ for all L_n and such that $E[g(L_n)]$ is easily calculated. Then we have the average-case bound $E[H(L_n)] \leq E[g(L_n)]$. A similar approach applies to the analysis of MS heuristics.

As a concrete example, we consider the LPT heuristic and its average idle time, as defined by (7.2). We have (Loulou, 1984; Frenk and Rinnooy Kau, 1987)

$$\begin{aligned} A^{LPT}(L_n, m) &\leq \mathrm{LPT}(L_n, m) - \sigma(L_n)/m \\ &\leq \max_{1\leq i\leq n} \left\{ X_{(i)} - \frac{1}{m} \sum_{k=1}^{i} X_{(k)} \right\}. \end{aligned} \tag{7.3}$$

To see the latter inequality, let i be the largest index such that $X(i)$ runs until the end of the schedule. Then just after $X_{(i)}$ is scheduled, the average processor idle time up to the end of the schedule is at most $(m-1)X_{(i)}/m \leq X_{(i)}$. Each task $X(k)$ scheduled after $X(i)$ reduces the average idle time by $X(k)/m$; (7.3) follows easily.

To illustrate the use of the bound (7.3), we show that (Frenk and Rinnooy Man, 1987)

$$A^{LPT}(L_n, m) \to 0 \text{ (a.s.) as } n \to \infty, \qquad (7.4)$$

when $E[X] < \infty$ and $F(x)$ is strictly increasing in (0, σ) for some σ > 0. Bounding the right-hand side of (7.3) by

$$X_{(\lceil \epsilon n \rceil)} + \max\left\{0, X_{(n)} - \frac{1}{m}\sum_{k=1}^{\lceil n \rceil} X_{(k)}\right\}, \qquad 0 < F^{-1}(\epsilon) < \delta, \qquad (7.5)$$

we observe that the first term in (7.5) converges (a.s.) to $F^{-1}(\epsilon) \text{ as } n \to \infty$ as and that it can be made arbitrarily small by an appropriate choice of ϵ. Also, since $E[X] < \infty$, $X_{(n)}/n \to 0$, (a.s.). Moreover, $\sum_{k=1}^{\lceil n\epsilon \rceil} X_{(k)}/n$ converges (a.s.) to a positive constant as $n \to \infty$ for every $\epsilon > 0$. Thus, the second tern within the maximization in (7.5) tends to $-\infty$ (a.s.). We conclude that (7.4) holds.

In some cases, the requirement that a bound hold deterministically for all L_n is too stringent to yield good results. In addition to a bound $H(L_n) \le g(L_n)$ that always holds, there may exist a sharper bound $g'(L_n)$ such that $H(L_n) \le g'(L_n)$ except on a set having a small probability q_n. If $q_n \to 0$ sufficiently rapidly that $q_n E[g(L_n)] = o(E[g'(L_n)]) \text{ as } n \to \infty$, then we have

$$\begin{aligned} E[H(L_n)] &\le (1 - q_n)E[g'(L_n)] + q_n E[g(L_n)] \\ &\sim E[g'(L_n)]. \end{aligned}$$

Dominating Algorithms. A common way to upper-bound $H(L_n)$ is to introduce a simpler, more easily analyzed algorithm H' for which it can be proved that $H'(L_n) \ge H(L_n)$ for all L_n. In this case, H' is said to **dominate** H. A similar approach applies to lower bounds. For example, the MATCH heuristic described below is dominated by both FFD and BFD.

The MATCH packing heuristic iterates the following procedure until all items are packed. Let S denote the set of items that remain to be packed. MATCH first finds a largest item in S, say X. If $|S| = 1$ or if no remaining item fits with X, i.e., $Y + X > 1$ for all $Y \in S - \{X\}$, then MATCH puts X into a bin alone. Otherwise, MATCH puts items X and X' into a bin alone, where X' is a largest remaining item other than X such that $X + X' \le 1$.

It can be proved without much difficulty that $\text{FFD}(L_n) \ge \text{MATCH}(L_n)$ and $\text{BFD}(L_n) \ge \text{MATCH}(L_n)$ for all L_n (Lueker, 1982). Moreover, MATCH has the following simple analysis when X ~ $U(0, 1)$. First, we have that $\mathrm{MATCH}(L_n) \leq (n+b)/2$, where b is the number of singleton bins in the MATCH packing. The number of singletons with an item no larger than 1/2 is at most one, so $\mathrm{MATCH}(L_n) \leq (n+b'+1)/2$, where b' is the number of singletons with an item larger than 1/2. But an inspection of MATCH shows that b' is equal in distribution to $\max_{1\leq i\leq n}\{\xi_i\}$, where ξ_i is a symmetric, n-step random walk starting at the origin. Standard results then yield $E[\mathrm{MATCH}(L_n)] = n/2 + \Theta(\sqrt{n})$ and hence $E[H(L_n)] = n/2 + \Omega(\sqrt{n})$, where H stands for either FFD or BFD.

7.2.3 Stochastic Planar Matching

Matching problems in one or more dimensions have arisen in the analysis of several packing heuristics. An example in one dimension was given in °7.2.2. Here, we first define a generalization of this matching problem to two dimensions and then illustrate how it occurs in the analysis of algorithms.

Let n plus points and n minus points be chosen independently and uniformly at random in the unit square. Let M_n denote a maximum **up-right matching** of plus points to minus points such that if a plus at (x, y) is matched to a minus at (x', y'), then $x \leq x'$ and $y \leq y'$. Let U_n denote the number of points left unmatched by M_n. The problem of determining the distribution of U_n is called the up-right matching problem. Asymptotic bounds on the expected value are given by (Leighton and Shot, 1986; Rhee and Talagrand, 1988; Shor, 1986)

$$E[U_n] = \Theta(\sqrt{n}\,\log^{3/4} n). \tag{7.6}$$

To illustrate the applications of (7.6), we consider the upper-bound analysis of the BF heuristic in Shor (1986), assuming that X ~ $U(0, 1)$. Define the **modified best fit (MBF)** heuristic to be the same as BF except that MBF closes a bin to any further items whenever the bin receives an item no larger than 1/2. Clearly, bins in an MBF packing have at most two items. It is not difficult to prove that MBF dominates BF, so that $E[\mathrm{BF}(L_n)] \leq E[\mathrm{MBF}(L_n)]$.

Next, we describe MBF as a matching procedure. Plot the items of L_n as points in the left half of the unit square so that X_i has a y coordinate 1-i/n and an x coordinate X_i if $X_i \leq 1/2$, and 1-X_i if $1/2 < X_i \leq 1$. X_i is plotted as a plus point if $X_i \leq 1/2$ and as a minus point if $1/2 < X_i \leq 1$. Now match a plus point with a minus point if the corresponding items are placed in the same bin by MBF. By definition of MBF, the minus point must be above the plus point, because the item corresponding to the minus point had to be scanned first. Also, the minus point must be to the right of the plus point, because the two items fit into a single bin. An MBF matching is a maximum up-right matching, as is easily verified. However, the model differs from the original one in two respects. First, points are samples in the left half of the unit square, and second, the *x* coordinate has been discretized so that $x \in \{0, 1/n, \ldots, (n-1)/n\}$. But it is easy to prove that (7.6) still holds; the effects of both differences are limited to changes in the hidden multiplicative constant.

Finally, we observe that MBF(L_n) is the sum of the occupied space σ(L_n) and the unoccupied space, the latter quantity being bounded by U_n. Thus, $E[\mathrm{MBF}(L_n)] = n/2 + \Theta(\sqrt{n}\log^{3/4} n)$, and hence

$$E[\mathrm{BF}(L_n)] = \frac{n}{2} + O(\sqrt{n}\log^{3/4} n)\,.$$

7.2.4 Linear Programming

If the item sizes in L_n, make up a discrete set, then BP is easily formulated as an integer program. Let s_1, ..., s_N be the different item sizes in L_n and let m_j, $1 \le j \le N$ be the number of items with size s_j. Define the *i*th possible **configuration** as a sequence of integers $C_{ij} \ge 0$, $1 \le j \le N$, such that $\sum_{j=1}^{N} C_{ij} s_j \le 1$, i.e., a set of items with C_{ij} of size s_j, $1 \le j \le N$, can be packed into a single bin. If *M* denotes the number of possible configurations, then $\mathrm{OPT}(L_n) = \sum_{i=1}^{M} t_i^*$, where $\{t_i^*\}$ solves the integer program: minimize $\sum_{i=1}^{M} t_i$ subject to $t_i \ge 0$, $1 \le i \le M$, and $\sum_{i=1}^{M} C_{ij} t_i = m_j$, $1 \le j \le N$.

Relaxations of such integer programs lead to useful bounds for the analysis of optimum solutions. For example, suppose we relax the integer program for L_n so that the t_i can be arbitrary nonnegative reals. Then it is readily shown that

$$\mathrm{LIN}(L_n) \le \mathrm{OPT}(L_n) \le \mathrm{LIN}(L_n) + N\,, \qquad (7.7)$$

where LIN(L_n) denotes a solution to the relaxed problem.

To illustrate the bound, consider the **packing constant**

$$c := \lim_{n \to \infty} E[\mathrm{OPT}(L_n)]/n\,.$$

We win show that for general *F*(*x*), $E[\mathrm{OPT}(L_n)] - nc = O(\sqrt{n})$ (Rhee and Talagrand, 1989b). This will also give us one of the many applications of Kolmogorov-Smirnov statistics to the analysis of BP and MS.

To begin, for some integer $N \geq 1$ to be chosen later, transform the given distribution *F* to a distribution *G* consisting of *N* atoms, each of weight 1/*N*, at $s_j = F^{-1}(j/N)$, $1 \leq j \leq N$. If c_N denotes the packing constant under *G*, then it is not hard to see that $c_N - 1/N \leq c \leq c_N$. Note that generating *n* items X_i according to *G* can be achieved by taking *n* uniform samples U_i from [0,1] and setting $X_i = F^{-1}([NU_i]/N)$, $1 \leq i \leq n$

To investigate c_N, consider the one-sided Kolmogorov-Smirnov statistic D_n^-, where

$$\begin{aligned} nD_n^- &= \max_{0 \leq z \leq 1} \{nz - |\{i : U_i \geq z\}|\} \\ &= \max_{0 \leq z \leq 1} \{|\{i : U_i > z\}| - (1-z)n\}. \end{aligned}$$

If we remove from L_n the items generated by the largest nD_n^- of the U_i and pack them one per bin, we are left with a list L_n' of items X_i' with sizes in $\{s_j\}_{1 \leq j \leq N}$ such that

$$|\{i : X_i' > s_j\}| \leq n(1-j/N), \quad 1 \leq j \leq N.$$

We have $\mathrm{OPT}(L_n') \leq \mathrm{OPT}(L_n'')$, where L_n'' contains exactly [*n/N*] items of each size s_j. Finally, let $\mathrm{LIN}(L_n'')$ denote the solution value of the LP relaxation for L_n'' in which we pack exactly *n/N* (rather than [*n/N*]) of each item size. By (7.7), we have $\mathrm{OPT}(L_n'') \leq \mathrm{LIN}(L_n'') + N$, and by the law of large numbers, we have $\mathrm{LIN}(L_n'') = Nc_N$. Hence we obtain the bound

$$\mathrm{OPT}(L_n) \leq nD_n^- + N + nc_N \leq nD_n^- + N + n(c + 1/N).$$

The standard bound, $E[nD_n^-] = O(\sqrt{n})$, along with the choice $N = \sqrt{n}$ then yields $E[\mathrm{OPT}(L_n)] - nc = O(\sqrt{n})$, where the hidden constant is independent of the distribution.

Duality theory has played a role in studies of the **perfect packing problem**: *For which distributions F do we have the packing constant* $c(F) = E[X]$, *so that* $E[\mathrm{OPT}(L_n)]/E[\sigma(L_n)] \to 1$ as $n \to \infty$? The dual of the integer program for BP is: *Find a set of nonnegative weights* u_j such that $\sum_{j=1}^{N} m_j u_j$ is maximized subject to $\sum_{j=1}^{N} u_j C_{ij} \leq 1$, $1 \leq i \leq M$. Note that the constraint simply requires that for any set of items fitting into a bin, the corresponding sum of weights must be at most 1.

The perfect packing problem was first studied within the class of uniform distributions *U*(*a, b*), $0 \leq a \leq b \leq 1$. Motivated by the dual problem above, a weighting function approach was adopted in Lueker (1983). We say that $u : [0,1] \to [0,1]$ is a **weighting function** if for all finite sequences quences x_1, ..., x_k of positive reals, $\sum_{i=1}^{k} x_i \le 1 \Rightarrow \sum_{i=1}^{k} u(x_i) \le 1$. It is easy to see that if *u* is a weighting function, then $\alpha(F) \ge E[u(X)]$. Using this relation and a computer program to help in the search, Lueker (1983) found weighting functions classifying the distributions *U* (*a, b*) such that $\alpha(U(a,b)) = E[X] = \mu$. For the general problem, Rhee and Talagrand (1989a, b) proved strong results drawing on ideas from functional analysis and topology. Courcoubetis and Weber (1986) studied the same problem within the class of on-line algorithms.

7.3 RELATED TOPICS

This section describes some of the more important questions that have grown out of the initial probabilistic studies of BP and MS.

7.3.1 Variants

The following problem has the same instance L_n as BP.

Bin Covering. *Partition* L_n into a maximum number of subsets (bins) such that each subset sums to at least 1.

The NF algorithm can be adapted to this problem in an obvious way. Applying standard results in renewal theory to the case $X \sim U(0, 1)$, Csirik et al. (1991) analyzed this variant of NF bin packing to obtain precise estimates of the expected number of bins covered.

The next problem has the same instance (L_n, m) as MS.

Dual Bin Packing. *Find a subset* $L'_n \subseteq L_n$ *of maximum cardinality* $C(L_n$, m) *such that* L'_n *can be partitioned into m subsets with each subset summing to no more than* 1.

Asymptotics for $E[C(L_n$, m)] have been studied in Bruno and Downey (1985).

7.3.2 Higher Dimensions

Extensions of BP to two and three dimensions have strong practical motivations, especially in stock-cutting applications. In the two-dimensional **strip**

packing problem, the rectangles of a list $L_n = (R_1, \ldots, R_n)$ are to be packed into a unit-width, semi-infinite strip; the heights and widths of all rectangles are at most 1. The packing is to have these properties: (1) the rectangles do not overlap each other or the edges of the strip, (2) rectangles are placed with their sides parallel to the edges of the strip (90° rotations are disallowed), and (3) the packing height is minimized, where the packing height is the maximum height reached by the tops of the rectangles in a vertically oriented strip.

In the variant, **two-dimensional bin packing**, horizontal boundaries are also placed at the integer heights of the vertical strip. Each rectangle must now be wholly contained within a unit square, or "bin," between some pair of consecutive integer heights. The objective is now to minimize the number of bins used in the packing.

The probabilistic analysis of two-dimensional packing has recently been surveyed in Coffman and Shot (1990). In the most common probability model, all rectangle heights and widths are taken to be independent samples from $U(0, 1)$. The heuristics studied have, for the most part, been straightforward extensions of one-dimensional algorithms. For example, any one-dimensional heuristic can be adapted to **level packings** of the strip, in which rectangles are placed along levels, or horizontal baselines. The first level is the bottom of the strip. Each higher level passes through the top of a highest rectangle in the preceding level. Thus, the space between adjacent levels corresponds to a one-dimensional bin. **Shelf packings** (Bartholdi et al., 1989) are similar except that levels are preset at heights determined by the distribution of rectangle heights, which is assumed to be given in advance. In general, the probabilistic analysis of level and shelf algorithms extends in natural ways the analysis of one-dimensional bin packing.

The two-dimensional bin packing algorithm in Karp et al. (1984) is a less obvious generalization of one-dimensional matching; its analysis reduces to that of up-right matching.

7.3.3 General Bounds

Lower bounds for BP have been useful in estimating the cost of certain restrictions to the design of algorithms. For example, assume $X \sim U(0, 1)$. Shor (1986) proved, as a result in stochastic planar matching, that

$$E[W^H(L_n)] = \Omega(\sqrt{n \log n})$$

for any on-line algorithm. (More recently, Shor (1991) devised an on-line algorithm that achieves this lower bound without knowing *n* in advance.) It is interesting to note, however, that if we augment the class of on-line algorithms by those that can make use of the number *n* of items to be packed, then this bound no longer applies; Shor (1986) devised an algorithm of this type that wastes $\Theta(\sqrt{n})$ space on average.

In another example, again under *U*(0, 1), algorithms have been classified by the number of active bins needed during the packing process. Here, an active bin is simply a nonempty bin that has yet to be closed to further items. It has been shown (Coffman and Shor, 1992) that if an algorithm *H* never has more than *r* bins active at any time, then $E[W^H(L_n)] \geq \frac{n}{16(r+1)}$. Note that $r = 1$ for NF. This result also applies to all other on-line, linear-time algorithms studied in the literature (e.g., Ramanan and Tsuga, 1992; Lee and Lee, 1987); i.e., all of these algorithms limit the number of active bins and produce a wasted space whose expected value grows linearly in *n*.

7.3.4 Distributions

In studies of BP the emphasis has been on the uniform distribution *U*(0, 1), because it leads to a tractable analysis. Models of MS have concentrated on *U*(0, 1) and exponential distributions, for the same reason. However, there have been many useful results dealing with more general distributions, as illustrated in §§7.2.2 and 7.2.4. For example, the lower bound on $E[\mathrm{OPT}(L_n)]$ under *U*(0, 1) is easily shown to apply to any symmetric distribution on [0,1]. These results have also been proved for distributions with decreasing densities, by a technique of decomposing such distributions into a series of symmetric distributions.

In addition to the perfect packing results alluded to in §7.2.4, there have been several cases where the uniform distributions *U*(0, *b*) have been successfully handled. Among these are Karmarkar's (1982) analysis of NF and the analysis by Bentley et al. (1984) of FFD. The analysis has often led to anomalous and unexpected results. For example, the analysis of FFD revealed discontinuities at $b = 1/2$ and $b = 1$, as shown by the fact that $E[W^{\mathrm{FFD}}(L_n)]$ is $O(1)$ if $0 \leq b \leq 1/2$, $(n^{1/3})$ if $1/2 < b < 1$, and as noted earlier, $\Theta(\sqrt{n})$ if $b=1$.

Quite recently, discrete uniform distributions have been studied in depth (Coffman et al., 1991). The results have shown that many aspects of average-case behavior are lost in the passage to continuous approximations $U(a,b)$. To illustrate, let $U\{j,k\}$ denote the uniform distribution on the set

$\{i/k\}_{1 \le i \le j}$, $1 \le j \le k$. Results for the continuous case do not suggest the following strong result: For any *U* {*j, k*} with $j \le k-2$, there is an *on-line* algorithm *A* such that $E[A(L_n)\text{-}\sigma(L_n)] = O(1)$. The behavior of classical packing rules has also been shown to be much more irregular in the discrete cases. For example, expected wasted space *O*(1) and (*n*) both occur under FFD for specific pairs *j,k* with *j/k* < 1/2 and with *j/k* > 1/2. As another example, there appear to be *j, k* such that FF produces *O*(1) expected wasted space, whereas FFD produces (*n*) expected wasted space.

7.4 DIRECTIONS FOR FURTHER STUDY

There are obvious open problems concerned with more general distributions *F*(*x*) and more precise results, e.g., useful bounds on the multiplicative constants hidden in the asymptotic notation. Here, we note a few gaps in the asymptotic theory that have yet to be resolved, even under the usual simplifying assumptions. We begin with two conjectures.

Conjecture 1 *For* $X \sim U(0, b)$, $0<b<1$, *we have* $E[W^H(L_n)] =$ (*n*) *for H = FF or BF.*

A similar conjecture applies to discrete distributions *U*{*j,k*} for $j \le k-2$.

The next conjecture refers to the expected processor idle time defined in (7.2). We again assume $X \sim U(0, 1)$.

Conjecture 2 *There exists an* $\alpha > 0$ *such that* $E[A^{\mathrm{OPT}}(L_n, 2)] = O(e\alpha n)$.

Strong results on the median of the distribution of $A^{\mathrm{OPT}}(L_n, 2)$ are proved in Karmarkar et al. (1986). The analysis applies the second moment method (Erdös and Spencer, 1974).

The conjectures for one-dimensional packing have their counterparts in higher dimensions. An interesting open problem in two dimensions is the average-case behavior of the FF and BF rules applied to the level algorithms of strip packing.

Industrial applications of three-dimensional packing abound, yet the design and probabilistic analysis of algorithms remain at an early stage (see Karp et al., 1984; Li and Cheng, 1990). For example, the existence of on-line algorithms with sublinear expected wasted space remains an open question.

REFERENCES

Bartholdi, J.J., J.H. Vande Vate, and J. Zhang (1989), Expected performance of the shelf heuristic for two-dimensional packing, *Oper. Res. Lett*. **8**, 11-16.

Bentley, J.L., D.S. Johnson, F.T. Leighton, C.C. McGeoch, and L.A. McGeoch (1984), Some unexpected expected behavior results for bin packing, in *Proc. 16th Annual ACM Symposium on Theory of Computing* , ACM Press, New York, 279-288.

Bruno, J.L., and P.J. Downey (1985), Probabilistic bounds for dual bin packing, *Acta Inf*. **22**, 333-345.

Coffman, E.G., Jr., and P.W. Shot (1990), Average-case analysis of cutting and packing in two dimensions, *Eur. J. Oper. Res*. **44**, 134-144.

Coffman, E.G., Jr., and P.W. Shor (1992), Packing in two dimensions: Asymptotic average-case analysis of algorithms, *Algorithmica*, to appear.

Coffman, E.G., Jr., C.A. Courcoubetis, M.R. Garey, D.S. Johnson, L.A. McGeoch, P.W. Shot, R.R. Weber, and M. Yannakakis (1991), Average-case performance of bin packing algorithms under discrete uniform distributions, in *Proc. 23rd A CM Symposium on the Theory of Computing*, ACM Press, New York, 230-241.

Coffman, E.G., Jr., G.N. Frederickson, and G.S. Lueker (1984a), Expected makespans for largest-first scheduling of independent tasks on two processors, *Math. Oper. Res*. **9**, 260-266.

Coffman, E.G., Jr., M.R. Garey, and D.S. Johnson (1984b), Approximation algorithms for bin packing—An up-dated survey, in *Algorithm Design for Computer System Design*, G. Ausiello, M. Lucertini, and P. Serafini, eds., Springer-Verlag, New York, 49-106.

Coffman, E.G., Jr., G.S. Lueker, and A.H.G. Rinnooy Kan (1988), Asymptotic methods in the probabilistic analysis of sequencing and packing heuris tics, *Manage. Sci*. **34**, 266-290.

Coffman, E.G., Jr., K. So, M. Hofri, and A.C. Yao (1980), A stochastic model of bin packing, *Inform. Contr*. **44**, 105-115.

Courcoubetis, C.A., and R.R. Weber (1986), Necessary and sufficient conditions for the stability of a bin packing system, *J. Appl. Probab* . **23**, 989-999.

Csirik, J., J.B.G. Frenk, A. Frieze, G. Galambos, and A.H.G. Rinnooy Kan (1986), A probabilistic analysis of the next fit decreasing bin packing heuristic, *Oper. Res. Lett*. **5**, 233-236.

Csirik, J., J.B.G. Frenk, G. Galambos, and A.H.G. Rinnooy Kan (1991), Probabilistic analysis of algorithms for dual bin packing problems, *J. Algorithms* **12**, 189-203.

Dyckhoff, H. (1990), A typology of cutting and packing problems, *Eur. J. Oper. Res*. **44**, 145-159.

Erdös, P., and J. Spencer (1974), *Probabilistic Methods in Combinatorics* , Academic Press, New York.

Feller, William (1971), *An Introduction to Probability Theory and Its Applications*, vol. II, John Wiley & Sons, New York, second edition.

Frenk, J.B.G., and A.H.G. Rinnooy Kan (1987), The asymptotic optimality of the LPT rule, *Math. Oper. Res*. **12**, 241-254.

Garey, M.R., and D.S. Johnson (1983), *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, New York.

Hofri, M. (1984), A probabilistic analysis of the next fit bin packing algorithm, *J. Algorithms* **5**, 547-556.

Hofri, M., and S. Kamhi (1986), A stochastic analysis of the NFD bin packing algorithm, *J. Algorithms* **7**, 489-509.

Johnson, D.S., A. Demers, J.D. Ullman, M.R. Garey, and R.L. Graham (1974), Worst-case performance bounds for simple one-dimensional packing algorithms, *SIAM J. Comput*. **3**, 299-325.

Karmarkar, N. (1982), Probabilistic analysis of some bin-packing algorithms, in *Proc. 23rd Annual Symposium on Foundations of Computer Science*, IEEE Computer Society Press, Los Alamitos, Calif., 107-111.

Karmarkar, N., and R.M. Karp (1982), The differencing method of set partitioning, Technical Report UCB/CSD 82/113, Computer Science Division (EECS), University of California at Berkeley, December.

Karmarkar, N., R.M. Karp, G.S. Lueker, and A.M. Odlyzko (1986), Probabilistic analysis of optimum partitioning, *J. Appl. Probab*. **23**, 626-645.

Karp, R.M., M. Luby, and A. Marchetti-Spaccamela (1984), A probabilistic analysis of multidimensional bin packing problems, in *Proc. 16th Annual A CM Symposium on Theory of Computing*, ACM Press, New York, 289-298.

Lee, C.C., and D.T. Lee (1987), Robust on-line bin packing algorithms, Technical Report, Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, Ill.

Leighton, F.T., and P.W. Shor (1986), Tight bounds for minimax grid matching with applications to the average case analysis of algorithms, in *Proc. 18th Annual ACM Symposium on Theory of Computing*, ACM Press, New York, 91-103.

Li, K., and K.-H. Cheng (1990), On three-dimensional packing, *SIAM J. Comput*. **19**, 847-865.

Loulou, R. (1984), Tight bounds and probabilistic analysis of two heuristics for parallel processor scheduling, *Math. Oper. Res*. **9**, 142-150.

Lueker, G.S. (1982), An average-case analysis of bin packing with uniformly distributed item sizes, Technical Report 181, Department of Information and Computer Science, University of California at Irvine.

Lueker, G.S. (1983), Bin packing with items uniformly distributed over intervals [*a, b*], in *Proc. 24th Annual Symposium on Foundations of Computer Science*, IEEE Computer Society Press, Los Alamitos, Calif., 289-297.

Ramanan, P., and K. Tsuga (1992), Average-case analysis of the modified harmonic algorithm, *Algorithmica*, to appear.

Rhee, W.T., and M. Talagrand (1988), Exact bounds for the stochastic upward matching problem, *Trans. Amer. Math. Soc*. **307**, 109-125.

Rhee, W.T., and M. Talagrand (1989a), Optimal bin packing with items of random sizes—II, *SIAM J. Comput*. **18**, 139-151.

Rhee, W.T., and M. Talagrand (1989b), Optimal bin packing with items of random sizes—III, *SIAM J. Comput*. **18**, 473-486.

Shor, P.W. (1986), The average-case analysis of some on-line algorithms for bin packing, *Combinatorica* **6**, 179-200.

Shor, P.W. (1991), How to do better than best-fit: Tight bounds for average-case on-line bin packing, in *Proc. 32nd Annual Symposium on Foundations of Computer Science*, IEEE Computer Science Press, Los Alamitos, Calif., 752-759.

8. Probability and Problems in Euclidean Combinatorial Optimization

J. Michael Steele,[1]
University of Pennsylvania

ABSTRACT

This chapter summarizes the current status of several streams of research that deal with the probability theory of problems of combinatorial optimization. There is a particular emphasis on functionals of finite point sets. The most famous example of such functionals is the length associated with the **Euclidean traveling salesman problem (TSP)**, but closely related problems include the minimal spanning tree problem, minimal matching problems, and others. Progress is also surveyed on (1) the approximation and determination of constants whose existence is known by subadditive methods, (2) the central limit problems for several functionals closely related to Euclidean functionals, and (3) analogies in the asymptotic behavior between worst-case and expected-case behavior of Euclidean problems.

No attempt has been made in this survey to cover the many important applications of probability to linear programming, arrangement searching, or other problems that focus on lines or planes.

[1] Research supported in part by NSF grant DMS88-12868, AFOSR grant 89-0301, ARO grant DAAL03-89-G-0092, and NSA grant MDA-904-H-2034.

8.1 INTRODUCTION

Almost all of the results surveyed here owe a debt of one sort or another to the classic theorem of Beardwood, Halton, and Hammersley that lays out the basic behavior of the length of the shortest tour through a random sample from a general distribution in $\mathbf{R}^d$:

Theorem 8.1 (Beardwood et al., 1959) *If* X_i, $1 \leq i < \infty$ *are independently and identically distributed random variables with bounded support in* $\mathbf{R}^d$, then the length L_n under the usual Euclidean metric of the shortest path through the points $\{X_1, X_2, \ldots, X_n\}$ *satisfies*

$$L_n/n^{(d-1)/d} \to \beta_{TSP,d} \int_{\mathbf{R}^d} f(x)^{(d-1)/d}\, dx \text{ almost surely.}$$

Here, f(x) is the density of the absolutely continuous part of the distribution of the X_i, and $\beta_{TSP,\,d}$ is a positive constant that depends on d but not on the distribution of the X_i.

The Beardwood, Halton, and Hammersley (BHH) theorem has led to developments of several different types. The first developments that we review concern generalizations of the BHH theorem that aim to provide insight into larger classes of processes. It turns out that the essential geometric features of the TSP are present in many of the problems that have been investigated in the theory of combinatorial optimization, and there is also a close connection between these properties and the notions of subadditivity. These subadditivity-driven generalizations of the BHH are addressed primarily in §8.2.

The second type of problem considered looks at the BHH theorem from the perspective of precision, rather than generality. It turns out that L_n, the length of the shortest tour through an *n*-sample, is highly concentrated about its mean. The main result of this development, the theorem of Rhee and Talagrand, is spelled out in §8.3. It tells us that the tails of the TSP functional L_n decay as rapidly as those of the normal distribution. This remarkable result has evolved through several intermediate stages, each of which offered new tools for the analysis of functionals of finite point sets in the plane.

Section 8.4 is the last to focus explicitly on the theory of the TSP, and, though we start to strain the notion of a Euclidean problem, the development casts light on problems like those just described. The main purpose of §8.4 is to introduce the connection between *Euclidean* optimization problems and self-similar sets of fractional dimension; the principal result reviewed is a theorem by S. Lalley on the TSP. This is an intriguing result that suggests a rich connection between the rapidly developing theory of sets of fractals and many classical problems.

In §§8.5 and 8.6 the set of problems studied is widened. We first consider the theory of the minimal spanning tree (MST) and review how one can construct an infinite analog to the MST that leads to answers of a number of classical questions. One of the best reasons for introducing any new mathematical structure is that it simplifies and strengthens earlier results, and this is exactly what occurs in this development of an "infinite MST." The new structure permits one to make many deductions that are apparently difficult (or perhaps impossible) to address without such a tool.

Section 8.6 gives a brief survey of some of the new developments in the theory of matching. This subject began with the theorem of Ajtai, Komlós, and Tusnády on the two-sample matching problem, but it has developed rapidly because of its rich connections to such topics as bin packing, scheduling, and the theory of empirical processes. There are many engaging special results in this active area, and it is not possible to offer much more than a taste. We particularly want to call attention to a new theory of majorizing measure initiated by M. Talagrand. This theory unifies and deepens many earlier results and also suggests many new concrete problems.

In the last three sections the pace is quickened, but the work that is surveyed represents many recent developments that can be expected to lead to much further research. In §8.7, recent progress is reviewed in the calculations of the constants that appear in results such as the BHH theorem. Section 8.8 then considers the very recent progress on the central limit problem for some geometric problems closely related to Euclidean functionals. Finally, in §8.9 we see some tight analogies between the probabilistic analysis of Euclidean functionals and their worst-case behavior. The concluding section mainly focuses on the omissions that have been occasioned by our focus on functionals on points rather than on the full range of Euclidean structures.

The Connection to Algorithms. Before going to the survey just outlined, there is one top-level observation about the BHH theorem that should be made. It is likely that this remarkable 1959 result would have remained relatively undeveloped if it had not been for the striking use of the BHH theorem in the polynomial-time probabilistic TSP algorithm of Karp (1976, 1977). The TSP was among the first of the traditional problems of geometric optimization to be proved to be NP-hard, and it is also a problem of genuine practical interest, so it is not surprising that considerable excitement was generated when Karp showed that the TSP is perfectly tractable under the plausible stochastic assumption that the sites to be visited by the tour can be modeled as a random uniform sample. In this context, Karp showed that, given any $\epsilon > 0$, there is a (simple!) polynomial time algorithm that produces a tour of length no more than $(1 + \epsilon)$ times the optimal tour length with probability that converges to one as the size of the problem is increased. This discovery launched an important branch of the field of probabilistic algorithms. Moreover, it generated powerful interest in the BHH theorem, its generalizations, and sharpenings that are at the center of this survey.

8.2 SUBADDITIVE EUCLIDEAN FUNCTIONALS

By abstracting from the traveling salesman tour just a few of its basic properties, it is possible to suggest a very general result that provides information comparable to that given by the BHH theorem for a large number of problems of combinatorial optimization in Euclidean space.

Let *L* be a function that associates a real number to each finite subset $\{x_1, x_2, \ldots, x_n\} \subset \mathbf{R}^d$. To spell out the most innocent properties of *L* that mimic the behavior of the TSP, we first note that for the TSP, *L* exhibits **homogeneity** and **translation invariance**; i.e.,

$$L(\alpha x_1, \alpha x_2, \ldots, \alpha x_n) = \alpha L(x_1, x_2, \ldots, x_n) \text{ for all } \alpha > 0\,, \qquad (8.1)$$

and

$$L(x_1 + y, x_2 + y, \ldots, x_n + y) = L(x_1, x_2, \ldots, x_n) \text{ for all } y \in \mathbf{R}^d. \qquad (8.2)$$

The TSP's length also has some strong smoothness and regularity properties, but these turn out not to be too important for most purposes. All that is needed is that *L* be Borel measurable when viewed as a function from $\mathbf{R}^{nd}$ to **R**. This condition is almost always trivial to obtain, but still one has to have it to be able to talk honestly about probabilities involving *L*.

Functions on the finite subsets of $\mathbf{R}^d$ that are measurable in the sense just described and that are homogeneous of order one and translation invariant are called **Euclidean functionals**. These three properties are commonplace but bland. One should not expect to be able to prove much in such a limited context, but, with the addition of just a couple of other structural features, a rich and useful theory emerges.

The first additional property of the TSP functional that we consider is that it is **monotone** in the sense that

$$L(x_1, x_2, \ldots, x_n) \leq L(x_1, x_2, \ldots, x_n, x_{n+1}) \text{ for } n \geq 1, \text{ and } L(\phi) = 0. \quad (8.3)$$

The final feature of the TSP functional that we abstract is the most substantial one. It expresses both the geometry of the underlying space and the fundamental suboptimality of one of the most natural TSP heuristics, the partitioning heuristic. If $\{Q_i\}_{i=1}^{m^d}$ is a partition of [0, t]^d into smaller cubes of edge length t/m, the **subadditive property** says there exists a constant B independent of m and t such that

$$L(\{x_1, x_2, \ldots, x_n\} \cap [0,t]^d) \leq \sum_{i=1}^{m^d} L(\{x_1, x_2, \ldots, x_n\} \cap Q_i) + Btm^{d-1} \quad (8.4)$$

for all integers $m \geq 1$ and real $t \geq 0$.

Euclidean functionals that satisfy equations (8.3) and (8.4) will be called **monotone subadditive Euclidean functionals**. This class of processes seems to abstract the most essential features of the TSP that are needed for an effective asymptotic analysis of the functional applied to finite samples of independent random variables with values in $\mathbf{R}^d$.

To see how subadditive Euclidean functionals arise (and to see how some problems just barely elude the framework), it is useful to consider two additional examples.

The first is the **Steiner minimum tree**. For any finite set of n points, $S = \{x_1, x_2, \ldots, x_n\} \subset \mathbf{R}^d$, a Steiner minimum tree for S is a tree T whose vertex set contains S such that the sum of the lengths of the edges in T is minimal over all such trees. Note that the vertex set of T may contain points not in S; these are called **Steiner points**. If $L_{ST}(x_1, x_2, \ldots, x_n)$ is the length of a Steiner tree of $x_1, x_2, \ldots, x_n$ and if we let $|e|$ be the length of an edge e, another way of defining L_{ST} is just

$$L_{ST}(S) = \min_T \{ \sum_{e \in T} |e| : T \text{ is a tree containing } S \subset \mathbf{R}^d,\ S \text{ finite} \}.$$

The second example is closely related, yet it points out that the innocuous monotonicity property of the TSP can fail even in natural problems. The example we have in mind is the **minimum spanning tree** . For

$\{x_1, x_2, \ldots, x_n\} \subset \mathbf{R}^d$, let $L_{MST}(x_1, x_2, \ldots, x_n) = \min \sum_{e \in T} |e|$, where the minimum is over all connected graphs T with vertex set $\{x_1, x_2, \ldots, x_n\}$. It is an easy matter to check that any minimizing graph must indeed be a spanning tree.

The functional L_{MST} is easily seen to be homogeneous, translation-invariant, and measurable. One can also check without much trouble that it is subadditive in the sense required above. Still, many simple examples such as the sets $S = \{(0,0), (0,2), (2,0), (2,2)\}$ and $S \cup \{(1,1)\}$ will show that L_{MST} fails to be monotone. One should suspect that this failure is of an exceptional sort that would not have great influence on asymptotic behavior, and this suspicion is well justified. Still, the example puts us on warning that non-monotone functionals can require delicate considerations that axe not needed in cases that mimic the TSP more closely.

The main theorem of this section shows that the properties (8.1) through (8.4), together with a modest moment condition, are sufficient to determine the asymptotic behavior of $L(X_1, X_2,..., X_n)$ when the X_i are independent and identically distributed:

Theorem 8.2 (Steele, 1981b) *Let L be a monotone subadditive Euclidean functional. If* $\{X_i\}$, i=1, 2, ... , *are independent random variables with the uniform distribution on* $[0, 1]^d$, and Var $L(X_1, X_2, \ldots, X_n) < \infty$ *for each* $n \geq 1$, *then as* $n \to \infty$

$$L(X_1, X_2, \ldots, X_n)/n^{(d-1)/d} \to \beta_{L,d}$$

with probability one, where $\beta_{L,d} \geq 0$ *is a constant depending only on L and d.*

The restrictions that this theorem imposes on a Euclidean functional are as few as one can reasonably expect to yield a generally useful limit theorem, and because of this generality the restriction to uniformly distributed random variables is palatable. Moreover, since many of the probabilistic models studied in operations research and computer science also focus on the uniformly distributed case, the theorem has immediate applications. Still, one cannot be long content with a theory confined to uniformly distributed random variables. Fortunately, with the addition of just a couple of additional constraints, the limit theory of subadditive Euclidean functionals can be extended to quite generally distributed variables.

To get to the essence of the extension, we first consider the case of random variables with a singular component, i.e., variables such that there is a measurable set $E \subset \mathbf{R}^d$ with Lebesgue measure zero such that $\mathrm{P}(X_i \in E) > 0$. Random variables with a singular component provide a kind of geometrical worst case and hence provide a useful test case.

To deal with such random variables, we need to draw out three additional properties shared by many subadditive Euclidean functionals. The first of these, called **scale-boundedness**, holds provided there is a constant C such that

$$L(x_1, x_2, \ldots, x_n) \leq Ctn^{(d-1)/d} \qquad (8.5)$$

for all $\{x_1, x_2, \ldots, x_n\} \subset [0, t]^d$. The second property, called **simple subadditivity**, holds provided there is a constant D such that

$$L(A_1 \cup A_2) \leq L(A_1) + L(A_2) + Dt \qquad (8.6)$$

for any finite subsets A_1 and A_2 contained in $[0, t]^d$.

These two properties hold for many natural examples, and in particular they are easily checked for the TSP. The key benefit of scale-boundedness and simple subadditivity comes from their showing that the singular component of a distribution makes a contribution that is of lower order than that of the absolutely continuous component.

Lemma 1 *Let L be a monotone subadditive Euclidean functional that is scale-bounded (8.5) and simply subadditive (8.6). If* $\{X_i\}$ *are independent identically distributed (i.i.d.) random variables and E is any bounded set of Lebesgue measure zero, then as* $n \to \infty$

$$L(\{X_1, X_2, \ldots, X_n\} \cap E)/n^{(d-1)/d} \to 0$$

with probability one.

The last property we will require for the extension of this theorem to general distributions calls on a type of restricted converse of simple sub-additivity. A Euclidean functional L is called **upper-linear** provided we have the following: For every finite collection of cubes Q_j, where $1 \leq j \leq M$ and the edges of the Q_j are parallel to the axes, and for every infinite sequence x_i, where $1 \leq i < \infty$ and $x_i \in \mathbf{R}^d$, we have

$$\sum_{j=1}^{M} L(\{x_1, x_2, \ldots, x_n\} \cap Q_j) \leq L(\{x_1, x_2, \ldots, x_n\} \cap \cup_{j=1}^{M} Q_j) + o(n^{(d-1)/d}). \qquad (8.7)$$

8.3 TAIL PROBABILITIES

The theory just outlined has a number of extensions and refinements. The first of these that we consider is the work of Rhee and Talagrand (1989) on the behavior of the tail probabilities of the TSP and related functionals under the model of independent uniformly distributed random variables in the unit *d*-cube. In Steele (1981a), it was observed that in dimension 2 Var (L_n) is bounded independently of *n*. This somewhat surprising result motivated the search for a more detailed understanding of the tail probabilities $P(L_n \geq t)$, and, in particular, it opened the question of determining if these probabilities might decay at the Gaussian rate $\exp(-ct^2/2)$.

After introducing several methods from martingale theory and interpolation theory that led to interesting intermediate results, Rhee and Talagrand (1989) finally provided a remarkable proof that in $d = 2$ the TSP (and many related functionals) do indeed have Gaussian tail bounds.

Theorem 8.3 (Rhee and Talagrand, 1989) *Suppose f is a Borel measurable function that assigns to each finite subset* $F \subset [0,1]^2$ *a real value f*(*F*) *such that*

$$f(F) \leq f(F \cup x) \leq f(F) + \min(\,d(x,y) : y \in F\,).$$

There is then a constant $K = K_f$ such that if X_i are independent and uniformly distributed in $[0, 1]^2$, *then the random variables defined by* $U_n = f(\{X_1, X_2, \ldots, X_n\})$ *satisfy*

$$P(|U_n - E(U_n)| > t) \leq \exp(-t^2/K)$$

The proof of this result rests on the systematic exploitation of the martingale versions of the Hoeffding and Prokhorov large-deviation inequalities together with a powerful bare-hands construction that helps articulate technical features of a random sample being well spread out.

8.4 THE TSP IN FRACTAL SPACES

The BHH theorem may seem like a result that is wedded to $\mathbf{R}^d$; certainly the growth rate $n(d\text{-}1)/d$ emerges as a natural characteristic of the dimension. Lalley (1990) has provided a set of results that cast new light on the role of dimension in problems like the TSP by developing limit results for TSP tours in fractal spaces. Lalley considered finite random samples chosen uniformly (in an appropriate sense) from self-similar subsets of $\mathbf{R}^2$ that have Hausdorff dimension strictly less than two. Lalley obtained an almost-sure limit law for the length L_n of the shortest tour through $\{X_1, X_2, \dots, X_n\}$, but now the normalizing denominator is no longer $\sqrt{n}$ but a power of n that depends on the structure of the self-similar set.

To give the flavor of the results obtained by Lalley, we will consider one specific example. A compact set K is said to be a strongly self-similar subset of $\mathbf{R}^2$ provided that there is a finite collection of transformations $T_i : K \to K$ so that

$$K = \bigcup_{i=1}^{m} T_i(K),$$

where each h T_i is a transformation that can be written as a *strict* affine contraction (with contraction factor $r_i < 1$) followed by a rigid motion. The strong self-similarity of K requires moreover that the intersections $T_i(K) \cap T_j(K)$, where $i \neq j$, be small in a technical sense that we will not detail.

Theorem 8.4 (Lalley, 1990) *If* X_i, $1 \leq i < \infty$, *are independent random variables with the "uniform distribution" on a strongly self-similar set If that has representation*

$$K = \bigcup_{i=1}^{m} T_i(K),$$

where $\|T_i(x) - T_i(y)\| = r_i\|x - y\|$, *then the length* L_n of the shortest tour through the point set $\{X_i : 1 \leq i \leq n\}$ *satisfies*

$$\lim_{n \to \infty} L_n/n^{\theta} = C_K$$

with probability one, where $C_K > 0$ *and* $0 < \theta \leq 1/2$ *is a constant uniquely determined by the similarity ratios* r_i.

8.5 MINIMAL SPANNING TREES

If x = $\{x_1, \dots, x_n\}$ is a finite set of points in $\mathbf{R}^d$ for $d \geq 2$, a **minimal spanning tree (MST)** t of x is a connected graph with vertex set x such that the sum of the edge lengths of t is minimal; i.e.,

$$\sum_{e \in t} |e| = \min_{G} \sum_{e \in G} |e|,$$

where $|e| = |x_i - x_j|$ is the Euclidean length of the edge $e = (x_i, x_j)$ and the minimum is over all connected graphs *G*.

Minimal spanning trees are among the most studied objects in combinatorial optimization, and even the probability theory of MST is rather well developed. For example, in Steele (1989) one finds results that cover the basic almost-sure asymptotic behavior of the MST that parallels the BHH theorem and even deals with edge weights that are more complicated than those given by the basic edge lengths. In particular, it is shown that if X_i are i.i.d. with compactly supported density *f* and 0 < α < *d*, then

$$n^{-(d-\alpha)/d}\sum_{e\in t}|e|^{\alpha} \to c(\alpha,d)\int_{\mathbf{R}^d} f(x)^{(d-\alpha)/d}\,dx \qquad \text{a.s.,}$$

where $c(\alpha, d)$ depends only on α and *d*.

One peculiar aspect of this result is that it fails to cover the extreme case α = *d*. This failure was particularly intriguing since the case α = *d* for *d* = 2 had been the source of an interesting conjecture by R. Bland, who by direct computational experience had been led to believe that if one takes the sum of the squares of the edges of the MST of a sample of points chosen randomly from the unit square, then as $n \to \infty$ the sum converges to a constant. This result is hinted at by the limit cited above, but the specifically desired result fell just outside of its domain.

In Aldous and Steele (1992) an approach to such limit problems was taken that differs completely from the subadditivity-based analyses such as that mentioned in §8.2 or the more ad hoc method used in Steele (1989). The new approach is based on the study of an analog of the MST for an *infinite* set of points, in particular the points of a Poisson process on all $\mathbf{R}^d$. The basic philosophy that guided the analysis is that because the empirical distributions of any independent uniform sequences look locally like a realization of the Poisson process, one should be able to relate the MST of a uniform sample to the MST of the (unbounded) Poisson process. Though this sounds a little like the "Poissonization" trick that goes back even as fax as Beardwood, Halton, and Hammersley, it is really radically different in concept and detail. To give a forward hint of that difference, note that we have no idea how to apply the method to the TSP. To sketch the basic idea, let x = (x_i) be a finite or countably infinite subset of $\mathbf{R}^d$, $d \geq 2$. We call x *nice* if (1) x is *locally finite*, i.e., has only finitely many elements in bounded subsets of $\mathbf{R}^d$, and (2) the interpoint distances ($|x_j - x_i|$, $i < j$) are all distinct. Now, given a pair (*x*, x) with x nice and $x \in \mathbf{x}$, we can define trees $t_m(x, \mathrm{x})$ with vertices from x as follows: Let $\xi_1 = x$, and let t_1

be the single vertex ξ_1. Let t_2 be the tree consisting of the vertex ξ_1 and the vertex $\xi_2 \in \mathbf{x} \setminus \{\xi_1\}$ that is closest to ξ_1 in Euclidean distance, together with the edge (straight line segment) connecting ξ_1 and ξ_2. We then proceed inductively in a greedy fashion and define $t_{\mathrm{m}} = t_{\mathrm{m}}(_{\mathrm{x}}, \mathrm{x})$ to be $t_{\mathrm{m-1}}$ together with a new edge (ξ_m, ξ_m), where $j_m \leq m-1$ and $\xi_m \in \mathbf{x} \setminus \{\xi_1, \ldots, \xi_{m-1}\}$ are chosen so that the edge length $|\xi_m - \xi_{jm}|$ is minimal (over all possible edges connecting $t_{\mathrm{m-1}}$ to x \ $t_{\mathrm{m-1}}$.) In the finite case where $n = |\mathbf{x}| < \infty$, this procedure terminates with the tree t_{n} (x, x), and in that circumstance the tree obtained does not depend on the choice of the starting vertex x.

When x is infinite, the situation is more complex. We write $t_\infty(x, \mathbf{x})$ for the set $\cup_n t_n(x, \mathbf{x})$, but some work is required to obtain useful structural information about this graph. In fact, it turns out to be technically useful to look at a different but related graph, described in the next lemma.

Lemma 2 *Let* $g = g(\mathrm{x})$ *be the graph on an infinite nice vertex set* x *defined by taking* (x_1, x_2) *as an edge in g if it is an edge in either* $t_\infty(x_1, \mathbf{x})$ *or* $t_\infty(x_2, \mathbf{x})$. *Then the graph g is a* **forest** *and each component of g is an infinite tree*.

The main object of Aldous and Steele (1992) is the random tree $\mathcal{T}$ constructed by taking a Poisson point process $\mathcal{N} = \{\eta_i\}$ of rate 1 in $\mathbf{R}^d$, letting $\mathcal{N}^0 = \mathcal{N} \cup \{0\}$. Let $\mathcal{G} = g(\mathcal{N}^0)$ be the forest constructed as in the lemma, and finally take the connected graph $\mathcal{T}$ to be the largest tree containing 0.

It is natural to conjecture that $\mathcal{G}$ is itself a *tree* with probability 1, but this possibility seems to be related to deep issues in continuum percolation, and the introduction of $\mathcal{T}$ permits one to finesse that subtle issue.

Lemma 3 *Let D be the degree of vertex* 0 *in* $\mathcal{T}$ *and let* L_1, ... , L_{D} be the lengths of the edges of $\mathcal{T}$ *incident to* 0. *We then have*

1. $D \leq b_d$, *a constant,*
2. $ED = 2$, *and*
3. $I_d \equiv \frac{1}{2}\sum_i EL_i^d < \infty$.

Now let $\mathcal{N}_n$ denote the point process consisting of n points $(\eta_i : 1 \leq i \leq n)$ that are independent and uniformly distributed on the unit cube $[0, 1]^d$. With probability 1, $\mathcal{N}_n$ is a nice subset of $\mathbf{R}^d$. Let $\mathcal{S}_n = t_n(\eta_1, \mathcal{N}_n)$ be the minimal spanning tree of these n vertices. It is intuitive that $\mathcal{S}_n$, after a suitable rescaling, should converge locally to $\mathcal{G}$.

Theorem 8.5 (Aldous and Steele, 1992) *(a) If* {|e_{i}|} *denotes the set of lengths of the edges of* S_n, *then*

$$\sum_{i=1}^{n-1} |e_i|^d \xrightarrow{L^2} I_d \text{ as } n \to \infty .$$

(b) If $\Delta_{n,i}$ denotes the proportion of vertices of S_n *with degree i, then for each i,*

$$E\Delta_{n,i} \to P(D = i) \text{ as } n \to \infty .$$

The existence of limits in (b) was proved by different methods in Steele et al. (1987), where one also finds a proof of almost sure convergence. The analogous question in the model in which the cost assigned to each edge is independent and identically distributed is solved in Aldous (1990) using "limit process" arguments such as those reviewed above.

8.6 MATCHING PROBLEMS

The problem of determining the least weight matching in a graph (*G, E*) for which there is a function that assigns a real value to the edges of *G* is a central problem of combinatorial optimization. The matching problem is intermediate in complexity between the MST problem and the TSP. The minimal matching problem cannot be solved by a naive greedy algorithm such as that used for the MST, but there is still a polynomial-time algorithm that does solve the problem—in contrast to the TSP.

There are two natural Euclidean versions of the matching problem. The simplest version considers 2*n* points in $\mathbf{R}^d$ and asks for the perfect matching of the set for which the total edge length is minimum. This matching problem has a theory that is almost perfectly parallel to the theory of the MST.

A more subtle set of issues arise when one instead considers two distinguished subsets $\{x_1, x_2, \ldots , x_{\mathrm{n}}\}$ and $\{y_1, y_2, \ldots ,y_{\mathrm{n}}\}$. Here the probability theory becomes much trickier, and new phenomena arise. A first taste of the new behavior is given by the results of Ajtai, Komlós, and Tusnády, one of which we note below:

Theorem 8.6 (Ajtai et al., 1984) *If* X_{i} and Y_{i} are independent and uniformly distributed in $\mathbf{R}^2$ *for* $1 \le i \le n$, *then there are constants* K_1 *and* K_2

such that

$$K_1\sqrt{n\log n} \le M_n = \min_{\sigma} \sum_{i=1}^{n} |X_i - Y_{\sigma(i)}| \le K_2\sqrt{n\log n}$$

with probability that approaches one as $n \to \infty$.

The $\sqrt{\log n}$ term in this result adds a new spin to the theory of Euclidean functionals for several reasons. First, it shows that the scaling arguments and self-similarity ideas that were useful for the TSP and the MST are no longer accurate enough to lead to the right orders. Still, the basic motivation remains substantially intact, and for $d \ge 3$ one again finds the characteristic growth rates of *n*(*d*-1)/*d*.

As an added impetus to the theory of Euclidean matching, there are some powerful connections with other parts of combinatorial optimization. It was through their work on bin packing and scheduling that Leighton and Shor were led to the investigation of the maximum length that must exist in a two-sample matching. The following example from their work again exhibits a curious logarithmic term:

Theorem 8.7 (Leighton and Shor, 1989) *If* X_i and Y_i are independent and uniformly distributed in $[0, 1]^2$ *for* $1 \le i \le n$, *then there is a constant K such that*

$$\min_{\sigma} \max_{1\le i\le n} |X_i - Y_{\sigma(i)}| \le K\sqrt{n}(\log n)^{3/4}$$

with probability that approaches one as $n \to \infty$.

In one of the most recent and most sweeping studies reviewed for this survey, Talagrand (1991) has unified the two preceding results as well as many others. The details of that development are too substantial to review here, but it is possible to indicate some of the sources of the new power.

Begin by noting that there is a basic relation between the behavior of the two sample matching problems and the theory of empirical discrepancy. We recall that if $\mathcal{C}$ is any class of functions from [0, 1] to [0, 1], the **empirical discrepancy** associated with $\mathcal{C}$ is defined by

$$D_n(C) = \sup_{f\in\mathcal{C}} \Big|\sum_{i=1}^{n} (f(X_i) - \int_{\mathbf{R}^2} f(x)dx)\Big|$$

where the X_i are i.i.d. random variables with density *f*. Even before the work of Ajtai, Komlós, and Tusnády, it was well understood that there was a close connection between M_n and D_n when $\mathcal{C}$ is taken to be the class of Lipschitz functions. It was also known that the theory associated with D_n was closely linked to the epsilon entropy of the class $\mathcal{C}$, and finally people were beginning to understand that the issues addressed by epsilon entropy could in some cases be more effectively addressed by an emerging method of majorizing measures.

Unfortunately, even defining all these terms would take us rather far afield, but hopefully it is meaningful to say that epsilon entropy is a tool for measuring the complexity of a class of functions by counting the number of epsilon balls that are needed to cover certain compact subsets, whereas the method of majorizing measures refines the theory of epsilon entropy to deal more effectively with aspects of inhomogeneity in $\mathcal{C}$. The substantial achievement of Talagrand (1991) was to develop the concrete tools of the theory of majorizing measures that make it directly useful in the theory of M_n and D_n and then to show how those tools could be used to deepen and unify the investigations initiated by Ajtai, Komlós, and Tusnády and broadened by Leighton and Shor.

8.7 THE VALUES OF THE CONSTANTS

There is a long history of effort devoted to the determination of the limiting constraints in results like the BHH theorem. There is also a substantial literature that addresses the constraints associated with worst-case upper bounds.

Work by Supowit et al. (1983), Moran (1984), and Goldstein and Rein-gold (1987) provide recent progress on the worst-case bounds and provide surveys of the older literature. Two particularly noteworthy contributions for the TSP are those of Goddyn (1990) and Karloff (1987). The latter broke the $\sqrt{2}$ barrier in dimension 2 by showing that $\alpha_{TSP,2} \leq 0.984\sqrt{2}$, whereas Goddyn (1990) obtained the best bounds in general dimension by means of a powerful technique that uses an infinite number of translations of quantizers other than cubical cylinders.

After the constraints on the TSP, the best-studied value is $\alpha_{RST,\ d}$, the constant associated with the worst-case length of a rectilinear Steiner minimum tree in the unit d-cube. Chung and Graham (1981) proved that αRST, 2 = 1, which is significant in that it is the only nontrivial worst-case constant for which we have an exact expression. The problem of determining $\alpha_{RST,\ d}$ in higher dimensions is still open, the bounds being

$\max(1, d/(4e)) \le \alpha_{RST,d} \le d4^{(1-d)/d}$ for $d \ge 1$ (cf. Snyder, 1990, 1991; Salowe, 1991).

Other than these bounds, little progress concerning the $\beta_{L,\ d}$ was made until Bertsimas and van Ryzin (1990) developed exact expressions for the probabilistic minimum spanning tree and matching constants as *d* gets large. Specifically, they showed that $\beta_{MST,d} \sim \sqrt{d/2\pi e}$ and $\beta_{M,d} \sim (1/2)\sqrt{d/2\pi e}$ as $d \to \infty$.

The crowning achievement concerning the probabilistic constants was the determination of an exact expression for $\beta_{MST,\ d}$ for all $d \ge 2$ by Avram and Bertsimas (1992). The expression for $\beta_{MST,\ d}$ is a series expansion, each term of which requires a rather difficult integration. Still, the first few terms of the series in dimension 2 have been computed to yield a numerical lower bound of $\beta_{MST,2} \ge 0.599$, which agrees well with experimental data. We note that the proof employed by Avram and Bertsimas relies strongly on the fact that a greedy construction is guaranteed to yield an MST. Unfortunately, such a construction fails for many objects of interest, including the TSP. It is therefore evident that entirely new methods probably will be required to develop exact expressions for constants such as $\beta_{TSP,\ d}$ that arise from NP-hard problems.

As a final note, we return to the rectilinear Steiner minimum tree problem. Bern (1988) showed that, for a Poisson process with intensity *N* on the unit square, the value of the rectilinear MST constant is at least that of the rectilinear Steiner tree constant plus 0.0014. This separation of the constants was increased by Hwang and Yao (1990), who also extended the result to include points uniformly distributed in the unit square.

8.8 THE CENTRAL LIMIT PROBLEM

The recent work of Avram and Bertsimas (1992) provides enticing progress on the important and long-standing problem of providing a central limit theory (CLT) for Euclidean functionals. Their method falls short of providing a CLT for the MST problem or the TSP, but it provides very useful CLTs for somewhat easier problems, including the *k*th nearest neighbor problem, the Delaunay triangulation, and the length of the Voronoi diagram.

There axe two keys to the approach used by Avram and Bertsimas. The first is the observation of the applicability of the relatively recent CLT for dependent random variables due to Baldi and Rinott (1989). This new CLT deals with the dependence relations within a collection of random vari ables through the structure of a dependency graph G_n and offers an explicit Berry-Essen type bound where the maximal degree $D_n G_n$ plays a critical role. The second key observation is that one achieves a considerable simplification of the CLT problem by first conditioning on the event A_n that in the subdivision of [0, 1] into subcubes of approximate size $n/\log n$ each subcube contains at least one and at most $e \log n$ of the sample points. The specifics of the problems handled by the Avram-Bertsimas method are such that once one conditions on A_n, one finds a problem that is within range of the Baldi-Rinott CLT. Any given problem has a number of details that must be directly resolved.

8.9 WORST-CASE GROWTH RATES

One engaging aspect of the asymptotic theory of combinatorial optimization of point functionals is the persistent similarity between the probabilistic rates of growth that have been reviewed and the behavior of the corresponding functionals in worst-case settings.

As a primary example of a worst-case growth rate, consider the **worst-case length** of an optimal traveling salesman tour in the unit *d*-cube:

$$\rho_{TSP}(n) = \max_{\substack{S \subset [0,1]^d \\ |S| = n}} \min_T \Big\{ \sum_{e \in T} |e| : T \text{ is a tour of } S \Big\}, \qquad (8.8)$$

In words, $\rho_{TSP}(n)$ is the maximum length over all n point sets in $[0, 1]^d$ that an optimal traveling salesman tour can attain. There is absolutely no probability theory here, for the point sets and tours are deterministic, yet there is a theory for this function that parallels the BHH theorem as closely as one could hope.

Theorem 8.8 (Steele and Snyder, 1989) *As* $n \to \infty$,

$$\rho_{TSP}(n) \sim \alpha_{TSP,d}\, n^{(d-1)/d},$$

where $\alpha_{TSP,d} > 0$ *is a constant depending only on the dimension d.*

The proof of this theorem and several related results all call on the subadditivity and smoothness properties of the underlying functional. The following elementary lemma often helps focus one's investigation of these problems by putting a finger on one clear set of sufficient conditions.

Lemma 4 *If* ρ(1) = 0 *and there exists a constant* $c \geq 0$ *such that for all integers* $m \geq 1$ *and* $k \geq 1$, *we have*

1. $\rho(n+1) \leq \rho(n) + cn^{-1/d}$ *and*
2. $m^{d-1}\rho(k) - m^{d-1}k^{(d-1)/d}r(k) \leq \rho(m^d k)$, ,

where $r(k) \to 0$ *as* $k \to \infty$, *then as* $n \to \infty$,

$$\rho(n) \sim \alpha n^{(d-1)/d}$$

for a positive constant α.

For the analysis of any particular problem, there is almost always serious work to be done to make the problem amenable to this lemma or its variant. Still, the lemma has already proved its effectiveness in a number of problems, including minimum matchings (Snyder, 1987), minimum spanning trees (Steele and Snyder, 1989), greedy matchings (Snyder and Steele, 1990), and rectilinear Steiner minimum trees (Snyder, 1991).

8.10 CONCLUDING REMARKS

In the areas in which this survey has been forced to make the most substantial omissions, there are good sources to which one can turn. One important class of results that we have touched upon concerns the study of discrepancies of sequences that are more general than random samples. This subject goes under the name of **irregularity of distribution**, and it is closely allied in many technical ways with the work surveyed here. The volume by Beck and Chen (1987) gives one a view of the subject that it would be wrong to try to reprise, though it would have been interesting to make more explicit some of the relationships to questions such as the MST and Steiner trees.

Another huge area of related problems that have not been engaged here fails in the domain of the recent book *Intersection and Decomposition Algorithms for Planar Arrangements*, by F.K. Agarwal (1991). The work of K. Clarkson, D. Dobkin, H. Edelsbrunner, L. Guibas, E. Welzl and many others is surveyed there, and the volume offers a compelling view of the power that emerges from a detailed understanding of the way a set of lines cut up the plane. Duality is one of the powerful tools of that subject, so the restriction that is made here to focus on problems involving only "sets of points" and not "sets of lines" would not show up as a valid distinction in the context of that theory. Still, a division had to be drawn, and there can be practical differences even where there are formal isomorphisms.

A third class of related results that have only been touched on here concern bin packing and its relation to scheduling. This topic is surveyed in Chapter 7 of this volume as well as in the beautiful recent book by Coffman and Lueker (1991) that is devoted to the topic.

Galileo is supposed to have said, "To study science, one must speak the language of science, and the language of science is written in circles, lines, and squares." With luck, this survey reveals that there is something a priori appropriate about the probability theory of combinatorial optimization. The subject has not strayed far from motivations that would have had meaning for Galileo. Still, the development has been extensive, and the methods have become powerful and diverse.

Acknowledgment. Sections 8.7 and 8.9 of this chapter are based in part on the geometrical portions of the article "Probability Theory of Network Optimization," written jointly with T.L. Snyder, which will appear in a multivolume series, *The Handbook of Operations Research* , coming from North-Holland Publishers. The author also thanks D. Aldous, D.F. Avram, D. Bertsimas, A.S. Golstein, L. Goddyn, E.M. Reingold, T.L. Snyder, J.S. Salowe, and M. Talagrand for making available prepublication copies of their work.

REFERENCES

Agarwal, F.K. (1991), *Intersection and Decomposition Algorithms for Planar Arrangements*, Cambridge University Press, New York.
Ajtai, M., J. Komlós, and G. Tusnády (1984), On optimal matchings, *Combinatorica* **4**, 259-264.
Aldous, D.J. (1990), A random tree model associated with random graphs, *Rand. Struct. Alg*. **1**, 383-402.

Aldous, D.J., and J.M. Steele (1992), Asymptotics of Euclidean minimal spanning trees on random samples, to appear in *Probability and Related Fields*.

Avram, F., and D. Bertsimas (1992), The minimum spanning tree constant in geometrical probability and under the independent model: A unified approach, *Ann. Appl. Probab*. **2**, 113-130.

Baldi, P., and Y. Rinott (1989), On normal approximations of distributions in terms of dependency graphs, *Ann. Probab*. **17**, 1646-1650.

Beardwood, J., J.H. Halton, and J. Hammersley (1959), The shortest path through many points, *Proc. Cambridge Philos. Soc*. **55**, 299-327.

Beck, J., and W.W.L. Chen (1987), *Irregularities of Distribution*, Cambridge University Press, New York.

Bern, M.W. (1988), Two probabilistic results on rectilinear Steiner trees, *Algorithmica* **3**, 191-204.

Bertsimas, D., and G. van Ryzin (1990), An asymptotic determination of the minimum spanning tree and minimum matching constants in geometric probability, *Oper. Res. Lett*. **9**, 223-231.

Chung, F.R.K., and R.L. Graham (1981), On Steiner trees for bounded point sets, *Geom. Dedic*. **11**, 353-361.

Coffman, E.G., Jr., and G.S. Lueker (1991), *Probabilistic Analysis of Packing and Partitioning Algorithms*, John Wiley & Sons, New York.

Goddyn, L. (1990), Quantizers and the worst-case Euclidean traveling salesman problem, *J. Comb. Theory B,* **50**, 65-81.

Goldstein, A.S., and E.M. Reingold (1987), Improved bounds on the traveling salesman problem in the unit cube, Technical Report, Department of Computer Science, University of Illinois, Urbana-Champaign, Ill.

Hwang, F.K., and Y.C. Yao (1990), Comments on Bern's probabilistic results on rectilinear Steiner trees, *Algorithmica* **5**, 591-598.

Karloff, H.J. (1987), How long can a Euclidean traveling salesman tour be?, Technical Report 87-20, Department of Computer Science, University of Chicago, Chicago, Ill.

Karp, R.M. (1976), The probabilistic analysis of some combinatorial search algorithms, in *Algorithms and Complexity: New Directions and Recent Results*, J.F. Traub, ed., Academic Press, New York, 1-19.

Karp, R.M. (1977), Probabilistic analysis of partitioning algorithms for the traveling salesman problem in the plane, *Math. Oper. Res*. **2**, 209-224.

Lalley, S.P. (1990), Traveling salesman with a self-similar itinerary, *Probab. Eng. Inform. Sci*. **4**, 1-18.

Leighton, T., and P.W. Shor (1989), Tight bounds for minimax grid matching with applications to the average case analysis of algorithms, *Combinatorica* **9**, 161-187.

Moran, S. (1984), On the length of optimal TSP circuits in sets of bounded diameter, *J. Comb. Theory B,* **37**, 113-141.

Rhee, W.T., and M. Talagrand (1989), A sharp deviation inequality for the stochastic traveling salesman problem, *Ann. Probab*. **17**, 1-8.

Salowe, J.S. (1991), A note on lower bounds for rectilinear Steiner trees, Technical Report, Department of Computer Science, University of Virginia, Charlottesville, Va.

Snyder, T.L. (1990), On minimal rectilinear Steiner trees in all dimensions, in *Proc. 6th Annual ACM Symposium on Computational Geometry* , ACM Press, New York, 311-320.

Snyder, T.L. (1991), Lower bounds for rectilinear Steiner trees in bounded space, *Inform. Process. Lett*. **37**, 71-74.

Snyder, T.L., and J.M. Steele (1990), Worst-case greedy matchings in the unit *d*-cube, *Networks* **20**, 779-800.

Steele, J.M. (1981a), Complete convergence of short paths and Karp's algo rithm for the TSP, *Math. Oper. Res*. **6**, 374-378.

Steele, J.M. (1981b), Subadditive Euclidean functionals and non-linear growth in geometric probability, *Ann. Probab*. **9**, 365-376.

Steele, J.M. (1989), Efficacy of spacefilling heuristics in Euclidean combinatorial optimization, *Oper. Res. Lett*. **8**, 237-239.

Steele, J.M., and T.L. Snyder (1989), Worst-case growth rates of some problems from combinatorial optimization, *SIAM J. Comput*. **18**, 278-287.

Steele, J.M., L.A. Shepp, and W.F. Eddy (1987), On the number of leaves of a Euclidean minimal spanning tree, *J. Appl. Probab*. **24**, 809-826.

Supowit, K.J., E.M. Reingold, and D.A. Plaisted (1983), The traveling salesman problem and minimum matchings in the unit square, *SIAM J. Comput*. **12**, 144-156.

Talagrand, M. (1991), Matching theorem and empirical discrepancy computations using majorizing measures, Technical Report, Department of Mathematics, Ohio State University, Columbus, Ohio.

9. Probabilistic Analysis in Linear Programming

Ron Shamir
Tel Aviv University[1]

ABSTRACT

The main results on probabilistic analysis of the simplex method and on randomized algorithms for linear programming are reviewed briefly.

9.1 THE LINEAR PROGRAMMING PROBLEM

The **linear programming (LP)** problem calls for minimizing a linear objective function subject to linear constraints. Every such problem can be presented in the following form:

$$\min\{ c^T x \mid Ax \geq b,\ x \geq 0 \}. \qquad (P)$$

Here the vectors $c^T = (c_1, \ldots, c_d)$, $b^T = (b_1, \ldots, b_m)$, and the $m \times d$ matrix A constitute the input, and $x = (x_1, \ldots ,x_d)$ is the vector of variables to be optimized. Usually, one assumes that all input numbers are integers or rationals. The goal is to find an optimal x if such exists, and if not, to determine that the problem is unbounded (i.e., the minimum is $-\infty$) or infeasible (i.e., there is no x satisfying all the constraints).

[1] This chapter was written while the author was a visitor at DIMACS and RUTCOR at Rutgers University. Supported by AFOSR grants 89-0512 and 90-0008 and by NSF grant NSF-STC88-09648.

LP is one of the fundamental problems in operations research and computer science. It was formulated in the 1940s by George Dantzig, who also proposed a method for solving it. Since then, the problem has been studied in thousands of scientific papers and hundreds of books, and interest in the problem has stayed on a very high level. One reason for this is the wide applicability of LP: it is used to model practical, real-life problems in economics, industry, communications, military, and numerous other areas. It is perhaps the most widely used optimization model in the world, and except data structures, perhaps the largest single use of computer resources (see Lovász, 1980.) Another reason is that LP problems with special structure arise in various theoretical and practical areas. Yet another reason is that some fundamental properties of LP and its algorithms are still not fully understood. One of these is the efficiency of the simplex method, which will be discussed here.

The method that was proposed by Dantzig for solving LP is the simplex method (e.g., Dantzig, 1963). To describe it geometrically, we need some terminology: A point in $\mathbf{R}^d$ is **feasible** if it satisfies all the constraints. The set of feasible points,

$$X = \{ x \in \mathbf{R}^d \mid Ax \geq b, \; x \geq 0 \},$$

is a polyhedron, i.e., the intersection of a finite set of half-spaces. Each half-space is of the form $\{x|A_{i*}x \geq b_i\}$, and its corresponding hyperplane is $\{x|A_{i*}x = b_i\}$. (A_{i*}. denotes the *i*th row in *A*.) For simplicity, we assume that the problem is **nondegenerate**; i.e., each set of *d* hyperplanes meet at a distinct single point. In that case, if $X \neq \emptyset$, the dimension of *X* is *d*, and each feasible point in which exactly *d* of the inequalities are satisfied as equalities is a **vertex** of *X*. Vertices are **adjacent** if they are connected by an edge of *X*, or, equivalently, if their defining sets of equalities have *d*-1 identical elements. It is not hard to see that if there is an optimal solution, its value will be obtained at some vertex of *X*. Assuming non degeneracy, the set of feasible points lying on one defining hyperplane is (if nonempty) of dimension *d* -1 and is called a **facet** of the polyhedron.

The simplex method starts from a feasible vertex, and moves in each iteration to an adjacent vertex, until eventually either an optimal vertex is reached or the algorithm recognizes at some vertex that the problem is unbounded. The move from one vertex to an adjacent one is called a **pivot**, and it can be realized mathematically by a simple manipulation of the constraints matrix. Usually, the next vertex is chosen so that the objective function value will improve after the pivot step. Even so, there is still a lot of freedom in choosing the next vertex. The specific rule of this choice is called a **pivoting rule**. Numerous pivoting rules have been devised for the simplex method, each giving rise to a different variant of the method. To obtain the initial feasible vertex, one can set up another LP problem (called the **phase I** problem) to which there is always a known feasible vertex, and apply the simplex method to that problem. This problem is set up in such a way that it is bounded, and when its optimal solution is reached, one either has a starting feasible vertex for the original problem or knows that that problem is infeasible. The scope of this chapter does not allow us to get into more details. We refer the interested reader to the (very incomplete) reference list. Also, several parenthetical comments on duality can be ignored by the reader who is not familiar with it. They are needed for accuracy but not for understanding the essence of the results.

Within a few years of its introduction, LP had become a central—perhaps the central—paradigm of operations research. The simplex method had proved itself to be extremely useful and highly efficient in practice. However, by the early 1970s it became clear that the simplex is not—as far as we know—a theoretically efficient algorithm: the complexity of the simplex method was shown to be finite, but not polynomial in the input length. In fact, on specially structured "bad" examples, many simplex variants were shown to require a number of pivots exponential in the problem dimensions. Hence, as far as we know, the simplex is *not* a good algorithm from the theoretical point of view. On the other hand, Monte Carlo studies and vast practical experience with real-life problems show that the typical number of pivots is very low: it appears to be a linear or slightly superlinear function of the dimensions. How can we explain the huge gap between the probably bad worst-case bounds and the efficiency of the method in practice? A natural approach is to apply probabilistic analysis: Assume that the input data are distributed according to some prespecified distribution and show that *on the average* the number of pivots is small.

This chapter gives a brief survey of various results from probabilistic analysis of the simplex method. The scope of the chapter forces us to omit many details, to say nothing of proofs. We shall be able only to give the flavor of the results and the different approaches, in the hope of kindling the reader's curiosity. The interested reader may find more detailed surveys in Borgwardt (1987), Todd (1991), and Megiddo (1987) and in Shamir (1987), from which this chapter originated. Much more on the context of the problem can be found in the references thereof.

9.2 PROBABILISTIC ANALYSIS

In order to perform probabilistic analysis, one has to specify the simplex variant used, the initialization (phase I) procedure, the form of LP used, and the probabilistic assumption on the distribution of problem inputs. Given those parameters, we denote by $\rho(n,d)$ (or $\rho(m, d)$) the average number of pivots for problems with n inequalities in d variables (or m inequalities in d nonnegative variables).

All the probabilistic models discussed here generate problems that are nondegenerate with probability one. To avoid being too technical, we do not describe the conditions that guarantee nondegeneracy in each model, but simply assume throughout that all the models deal only with (primal and dual) nondegenerate problems.

9.2.1 Parametric Simplex Variants

The variant of the simplex that has been proved most amenable to probabilistic analysis is the parametric objective simplex algorithm, due to Gass and Saaty (1955). This variant—explicitly or in disguise—plays a central role in most of the studies discussed below. Let us sketch the parametric simplex method and introduce some terminology that is needed later. Suppose we have a LP with two objective functions, the original objective c and a *co-objective* c'. Given an optimal solution to problem (P) with objective c', the goal is to solve problem (P) with the original objective c. The problem can be presented as one with a parametric objective:

$$\min\{\lambda c^T x + (1-\lambda)c'^T x \mid Ax \geq b,\ x \geq 0\}. \qquad (P(\lambda))$$

Here λ is a scalar parameter, a solution for $P(0)$ is known, and we want to solve problem $P(1)$. In the parametric simplex algorithm, the value of λ is gradually increased. A new problem is obtained for each λ, but for a certain range of λ values the optimal vertex remains the same. When the current optimal vertex ceases to be optimal, a pivot step is performed and the algorithm moves (along an edge) to a new vertex that is optimal for larger λ The process terminates when the value $\lambda = 1$ is reached or when the solution to the problem becomes unbounded at some critical value $\lambda < 1$, in which case problem $P(1)$ is unbounded. Under our blanket assumption of nondegeneracy, the set of optimal points visited by the algorithm is a path of vertices and edges. This set,

$$\{x \in \mathbb{R}^d \mid x \text{ is optimal for some } P(\lambda),\ 0 \leq \lambda \leq 1\},$$

is called the **efficient path**. Hence the number of vertices along that path equals the number of pivot steps performed by the parametric simplex algorithm. Another set of points can be obtained by the same algorithm when the value of λ decreases from 0 to $-\infty$. The solution for $\lambda = -\infty$ corresponds to the solution for maximizing (rather than minimizing) c^T x. The set of solutions for $-\infty < \lambda \leq 1$ is called the **co-optimal path**.

A similar idea can be applied to the following problem, in which b and c are parameterized simultaneously:

$$\min \{ \lambda c^T x + (1 - \lambda) c'^T x \mid Ax \geq \lambda b + (1 - \lambda) b', \ x \geq 0 \} . \qquad (\Pi(\lambda))$$

The **parametric self-dual (PSD)** simplex algorithm, which was introduced by Dantzig (1963), starts from an optimal solution to Π(0). It increases λ gradually and performs a sequence of (primal or dual) pivots that yield solutions for Π(λ) for increasing values of λ. Note that with an initial choice of $c'^T = (1, 1,. . .,1)$ and-$b'^T = (1, 1,. . .,1)$, x=0 is an optimal solution for Π(0), so this algorithm avoids the need for two simplex phases.

The convenience in analyzing the parametric simplex is due to its property that, given a vertex of the feasible set, one can determine directly if the simplex path passes through that vertex. "Directly" means by an algebraic closed-form condition depending on the vertex and the input data, so the pivots need not be performed.

9.2.2 Borgwardt's Results

The first major probabilistic analysis results were due to Karl Heinz Borgwardt (1977a,b; 1978). He analyzed problems of the form

$$\min \{ c^T x \mid Ax \leq e \} ,$$

where A is of dimension $n \times d$ and $e^T = (1, 1, \ldots, 1) \in \mathbf{R}^n$. Note that such problems are always feasible, since $x = 0$ is feasible. The basic probabilistic model he used requires that each of the n vectors constituting the rows of A and the vector c assumes independently the same spherically symmetric distribution on $\mathbf{R}^d$-{0}. In other words, the distribution of each such vector is invariant under any rotation around the origin. One can visualize such a distribution as a "cloud" around the origin, with the density of the cloud at each point proportional to the probability of choosing the vector from the origin to that point. In a spherically symmetric distribution, the density at each point on any sphere centered at the origin is the same, but spheres of different radii may have different densities. Hence the radial part of the distribution is the only part that may vary among such distributions.

Borgwardt used the parametric simplex. Borgwardt (1978) showed that when *c'* is also chosen randomly and independently according to the same distribution, the expected number of pivots along the efficient path for *d* fixed and $n \to \infty$ is asymptotically bounded by

$$\text{const} \cdot d^2 n^{1/(d-1)} .$$

A crucial step in Borgwardt's analysis was to restate the problem in terms of the polar set X^* of the feasible set *X*, which is also a polyhedron (cf. Grünbaum, 1967). The number of vertices along the efficient path equals the number of faces of X^* intersected by a two-dimensional cone spanned by the two objectives *c* and *c'*. Borgwardt then estimates the expectation of that number by an integral expression, which eventually leads to the results.

Borgwardt (1977a,b) also obtained interesting asymptotic upper and lower bounds for several specific spherically symmetric distributions. For example, when all the mass of the distribution is on the unit sphere, he obtained a lower bound of $d^{1.5}n1/(d\text{-}1)$ on the expected number of pivots. He also showed that, whenever *d* is fixed and $n \to \infty$, $\rho(n, d) \to \infty$ for every distribution with bounded support and that there exist distributions for which ρ(*n, d*) grows slower than any polynomial in *n*.

Later, Borgwardt (1981, 1982) obtained a nonasymptotic bound on the expected length of such an efficient path, namely,

$$\text{const} \cdot d^3 n^{1/(d-1)} .$$

In order to obtain a result for a complete (two-phased) simplex method, Borgwardt defined a new algorithm. It generates a sequence of subproblems with increasing numbers of variables and uses the parametric algorithm to solve each subproblem. Borgwardt (1982) proves a polynomial expected bound for that algorithm. This was the first proof that the average number of pivots of any complete simplex variant is polynomial. The bound for Borgwardt's complete algorithm and model was subsequently improved (Borgwardt, 1987) and now stands at

$$\rho(n, d) \leq \text{const} \cdot (d + 1)^4 n^{1/(d-1)} . \qquad (B)$$

Recently, K.H. Borgwardt (private communication, 1992) has improved the *asymptotic* analysis for the complete algorithm (i.e., phase I plus phase II) to const · $d^{2.5}n1/(d\text{-}l)$.

Because of the importance of Borgwardt's work, this section concludes by noting three problems in his model, which may be improved upon in the future: First, the model generates only feasible problems, for which a feasible point is known in advance. Second, the complete algorithm that was "tailored" to facilitate the analysis can be used only on problems whose form fits the model. (For some progress on these two problems, see Borgwardt (1990).) Third, its phase I solves a sequence of *n*-1 problems to get a feasible point for phase II, although such a point is known a priori. For experimental studies on the model described above, see Borgwardt et al. (1990).

9.2.3 Asymptotic Results: Smale and Others

Smale (1983a,b) investigated linear programs of the form (*P*). The probabilistic model in Smale (1983a) requires that the vector (c^{T},-b^{T}) assume a spherically symmetric distribution in **R**.*d*+*m*-{0}, and *A* assumes a spherically symmetric distribution in **R***m*×*d*-{0}.

The variant analyzed was the PSD simplex. Smale used a theory developed by Eaves and Scarf (1976) that implies a presentation of the PSD algorithm for problem Π(λ) as a set of piecewise-linear equations. Using this presentation, Smale (1983a) showed how to express the probability that a vertex is in the PSD-path in terms of the volume of a corresponding cone. A series of estimates for the volume of the special cone structure generated for LP eventually yielded the result

$$\rho(m,d) \le c(d) \cdot (1 + \log(m+1))^{d(d+1)} . \qquad (S)$$

In particular, this result implies that when *d* is fixed, ρ(*m, d*) grows with *m* more slowly than any fixed positive power of *m*. The dependence on *d* is, however, exponential.

Smale's model seems to have a certain advantage over Borgwardt's, because it uses a more practical variant and there is no restriction on the form of the linear programs it can solve. Also, the model generates both feasible and infeasible problems. However, the asymptotic results assume that *d* is fixed and $m \to \infty$; in that situation most problems generated under this model become infeasible, so it is not clear how to interpret such results. We shall return to this point below.

Following Smale, Blair (1986) showed that essentially the same asymptotic result can be obtained by a simpler argument, under somewhat weaker probabilistic assumptions. Smale's estimate (*S*) is meaningful only when *d* is fixed and $m \to \infty$. Blair approached that case directly. One way to restate his observation is that when $m \gg d$, under Smale's probabilistic model, most of the constraints are "dominated" by other constraints and cannot participate in vertices of the PSD path. (Intuitively, a constraint is dominated if another constraint separates it from the origin in the positive orthant. We omit the formal definition of domination here.) By obtaining an upper bound of the expected number of undominated constraints, he proved that for fixed *d* and $m \rightarrow \infty$

$$\rho(m,d) \leq c(d) \cdot (\log m)^{d(d+1)\log_2(d+1)+d} .$$

The exponent of log *m* is higher than in (*S*), but the result is essentially the same. Namely, for fixed *d*, ρ (*m, d*) grows with *m* slower than any polynomial. Blair also showed that the same result holds for two other simplex variants.

Adler et al. (1986) investigated several probabilistic models that generalize Smale's. Their most general model requires for LP of the form (*P*), in addition to some nondegeneracy assumptions, that the data distribution is invariant with respect to reversing the signs of any subset of the rows of the matrix (*A, b*). They define a family of algorithms that proceed according to a "constraint-by-constraint" (CBC) principle. This principle requires that vertices that satisfy as equality the *k*th constraint may be considered only if the subproblem defined by the *k*-1 preceding constraints has been shown to be feasible. (A dual form of the CBC method is closely related to the phase I given in Borgwardt (1982).) This facilitates the detection of infeasibility of a problem at a relatively early stage in order to exploit the high probability that a problem is infeasible under this model when $m \gg d$.

Using the CBC principle, the authors show that even if *full enumeration* of both feasible and infeasible intersection points of *d* hyperplanes is done in every subproblem,

$$\rho(m,d) \leq \mathrm{const} \cdot 2^{M} ,$$

independent of *m*. In particular, when *d* is fixed and $m \rightarrow \infty$, the expected number of steps is bounded by a *constant*. By using more efficient CBC algorithms and stronger probabilistic assumptions, the upper bound is reduced to a smaller function of *d*, but it still remains exponential. The authors also show that these results hold for a broad family of simplex variants.

Although the bound obtained above is better than in Smale (1983a), it is not an improvement on Smale's, because the variant that he analyzed is not a CBC algorithm. Using the approach developed in Smale (1983a), Megiddo (1986a) improved the analysis for Smale's model to

$$\rho(m,d) \le \bar{c}(d).$$

Furthermore, Megiddo showed that ρ>(*m, d*) decreases to the limit for any fixed *d*. However, $\bar{c}(d)$ is superexponential in *d*.

The results of Blair (1986) and Adler et al. (1986) emphasize a considerable drawback of the asymptotic analyses of sign-invariant (and spherically symmetric) models: Blair showed that Smale's result can be explained in terms of the small expected number of vertices of the feasible set, without really using the properties of the specific parametric simplex variant. Adler et al. showed that upper bounds for ρ(*m, d*) that depend on *d* only cam be obtained by *enumeration* procedures, without using any property of the simplex method. Both of these results indicate that those asymptotic bounds reflect the properties of *the probabilistic model* rather than those of the simplex method. Later studies, as we shall soon see, overcame this drawback by obtaining results that are meaningful for all *d* and *n*.

Later, Smale (1983b) extended his results to a more general probabilistic model. The main addition in that model is the replacement of spherical symmetry with invariance of the probability distribution under coordinate permutations of $\mathbf{R}^d$. The results of Adler et al. (1986) and Megiddo (1986a) do not apply to the more general model. Blair's result does apply to Smale's second model, and it indicates a similar drawback in interpreting asymptotic results under it.

9.2.4 Adler, Haimovich: Sign-Invariant Model for Phase II

Adler (1983) and Haimovich (1983) studied independently the expected number of pivots along paths generated by some parametric simplex variants. We present their results with respect to the form *P*(λ). The probabilistic model they used assumes **sign invariance:** The input distribution must be invariant under changing the signs of any subset rows and columns of

$$\begin{pmatrix} c^T & 0 \\ c'^T & 0 \\ A & b \end{pmatrix}.$$

(Changing the sign of row *i* simply replaces the inequality $A_{i*}x \le b_i$ by $A_{i*}x \ge b_i$. Changing the sign of column *i* is equivalent to changing the constraint $x_i \ge 0$ to $x_i \le 0$. In other words, the hyperplane does not change; rather, the other half-space that is supported by it is chosen.) In addition, a certain nondegeneracy condition is required. This model generalizes Smale's first model, since every spherically symmetric distribution satisfies the sign-invariance condition. Certain discrete distributions and certain problems with sparse matrices also fall under that model.

The main advantage of this model is that it facilitates analysis in terms of simple counting arguments. Under this model, Adler and Haimovich showed that the average number of pivots along a nonempty co-optimal path in a feasible LP is no more than

$$\min(m,d)+1\,.$$

Similar results were obtained using other parameterizations. In particular, Adler showed the same bound for the PSD simplex with the sign invariance assumptions extended to the additional right-hand side vector *b'*.

The Adler-Haimovich results were very significant but still less than a complete explanation of the good real-life behavior of the simplex method. These results show that the co-optimal path from a point that minimizes a random objective to a point maximizing that objective is on average short. One problem is that this analysis assumes that a phase II algorithm starts from a random point, and this assumption has not been proved so fax for any known variant. Furthermore, the co-objective must be chosen independently of the particular problem. In known phase I algorithms the starting point is not random but is either fixed or dependent on the particular data. Such differences may change the results dramatically. There are striking examples from linear complementarity theory (Megiddo, 1986b; Saigal, 1983) where a path starting from a random point has expected polynomial length and a path starting at a fixed point has expected exponential length.

9.2.5 The Quadratic Results

Toward the end of 1983, three independent investigations obtained an upper bound on ρ(*m, d*) that is *quadratic* in min(*m, d*). Two of the studies, in Todd (1986) and Adler and Megiddo (1985), analyzed the PSD simplex. The third, Adler et al. (1987), analyzed a parametric version of the constraint-by-constraint method. The probabilistic model was the same in all three investigations. It was essentially the sign-invariance model described in the previous section, with a stronger non degeneracy assumption. It is still more general than the model in Smale (1983a).

The PSD simplex was analyzed under a specific choice of so-called lexicographic initialization vector. Todd analyzed that algorithm using tools from matroid theory, whereas Adler and Megiddo developed further Smale's approach. Adler et al. analyzed a parametric CBC algorithm with a lexicographic co-objective, building on the previous results and ideas of Adler and Haimovich.

After the completion of these three investigations, Megiddo (1985) observed that, although the parametric CBC algorithm and the PSD algorithm are in general quite different, their lexicographic versions execute exactly the same sequence of pivots. Thus, all three investigations were concerned with *the same simplex variant*, and obtained the result

$$\rho(m,d) \leq \mathrm{const} \cdot [\min(m,d)]^2 .$$

Later, Adler and Megiddo (1985) obtained a quadratic lower bound on the average number of pivots. The probabilistic assumptions required for that result are stronger. They require that all coordinates of the data (*A, b, c*) are independent, identically distributed random variables with a common distribution symmetric about the origin. Their result was obtained by establishing bounds on the correlation between the probabilities for bases to be both primal and dual feasible simultaneously. That result, with the above, establishes that, at least for the stronger model, $\rho(m, d)$ is in fact bounded between two quadratic functions of $\min(m, d)$. This also proves that there cannot exist a subquadratic upper bound for the weaker model.

Unlike the Adler-Haimovich results, the quadratic bound was not obtained under conditioning of feasibility. The sign-invariance model manifests the same behavior as Smale's spherically symmetric model; namely, as the ratio n/d grows, the proportion of feasible problems diminishes, and the meaning of averaging the number of pivots over all feasible and infeasible problems is put in question. However, using the fact that the quadratic resuit holds for all n and d, it was shown that for $n = 2d$ the expected number of pivots in a feasible problem under that model is at most const · $d^{2.5}$.

9.3 RANDOMIZED ALGORITHMS

Suppose we allow a simplex algorithm to choose randomly among several possibilities at certain points along its computation. For example, the algorithm can randomly choose the next vertex to move to among the adjacent vertices that provide a better value to the objective. We can then compute the expected number of pivots required to solve a problem, when the average is taken over the possible random choices. The expected number of pivots of such a randomized algorithm is the largest average number of pivots on any problem with the same size. (The maximum is over all problems of this size, and the average is over the internal randomizations performed by the algorithm.) Hence in this approach the algorithm is randomized, and the input data are not. In contrast, in the probabilistic analyses described in previous sections the algorithm is completely deterministic, and the randomization is over the problem set.

Randomized pivoting rules have been suggested in the past for LP; see Dantzig (1963). Recently, there were several important developments in the theory of randomized algorithms for LP. Dyer and Frieze (1989) and Clarkson (1991) described a randomized algorithm for LP in case the dimension d is fixed, improving the deterministic algorithm of Megiddo (1984) for this special case. Clarkson's algorithm has expected complexity O (d^2n + dd/2+O(1) log n). Clarkson's method was also exploited to obtain a fast parallel algorithm for LP in fixed dimension (Alon and Megiddo, 1990) and generalized to convex programming (Adler and Shamir, 1989). Seidel (1990, 1991) described another very simple "dual simplex" randomized algorithm for LP in fixed dimension, of complexity (d!n). This has been improved by Sharir and Welzl (1992) and extended to cover a fairly general class of convex programming problems, to a dual simplex algorithm with $O(n \cdot 2^d)$ expected number of pivots. A previous attempt at improving Seidel's algorithm is due to Welzl (1991). It seems to run very fast in practice, but no analysis of its complexity is known yet.

In a remarkable recent development; Kale (1992b) has given the first randomized simplex algorithm for general LP problems that is subexponential. His algorithm takes on the average

$$n^{C \cdot \sqrt{d/\log d}}$$

pivots, where C is an absolute constant. Roughly speaking, his algorithm starts from a vertex v, then reaches vertices on r different facets, randomly chooses one facet among them, and works recursively on that facet to find the optimal vertex in it. A judicious choice of r (and a careful analysis) leads to the above bound.

Although this bound is not polynomial, this is a substantial improvement over the exponential upper bounds known previously. Kalai's work was preceded by important results on the diameter of polyhedra. (See Kalai (1991, 1992a) and Kalai and Kleitman (1992) and the excellent survey by Klee and Kleinschmidt (1987) for the context of this problem. In that context, Kalai has also shown that if one does not limit the computational effort per pivot step, *n*log *d* pivot steps always suffice.) Shortly after Kalai's result, Matoušek et al. (1992) showed that a careful analysis of the previous randomized simplex algorithm of Sharir and Welzl (1992) leads to the same expected complexity bound.

9.4 THE ROAD AHEAD

In retrospect, one can see two parallel lines of research on probabilistic simplex analysis. The first line of research, on the feasible rotation-symmetric model, was carried out single-handedly by Borgwardt. The second started with Smale's model and was later generalized to the sign-invariant model, leading finally to the quadratic results. In both models the analysis advanced in three stages:

1. asymptotic results for fixed dimension,
2. nonasymptotic results for phase II, starting from a vertex optimal with respect to a random objective, and
3. results for a complete simplex algorithm.

In both cases the average complexity and the "naturalness" of the variant had to be sacrificed to some extent between stages 2 and 3.

Although the gap between the good practical behavior of the simplex and its proven bad worst-case bounds has been explained in part by the probabilistic analyses, many related questions are still open. Both models have some undesirable properties that were mentioned above. Analyzing, or even merely defining, a widely acceptable model that generates "real-life" problems is a major challenge. Any analysis of the concentration of the number of pivots around the mean (e.g., the variance) will be new and interesting. More natural simplex variants, especially those used in practice, are also still awaiting analysis.

A short time after the quadratic results were obtained, Karmarkar (1984) announced a new algorithm for LP that is both theoretically and practically efficient. The interest of the research community shifted to this exciting development, and for a while some people believed that the simplex method has met its match and was now obsolete. Now that most of the dust has settled, it looks as if both algorithms have their respective advantages, and each has problems that it solves faster. Moreover, usually it is not possible to tell a priori which algorithm will be faster on a given problem. (One of the wonderful side benefits of this development is that the existing codes for the simplex method have been improved by almost two orders of magnitude to make them competitive with the new challenger.) As is often the case, new solutions raise new problems, and one of them is related to our theme: the efficiency of Karmarkar's method in practice is much faster than what can be proved theoretically. Explaining this gap by probabilistic analysis is a new great challenge for the LP community.

In randomized algorithms for LP, a major challenge is now to find a randomized simplex algorithm that is polynomial, or to prove that no such algorithm exists.

REFERENCES

Adler, I. (1983), The expected number of pivots needed to solve parametric linear programs and the efficiency of the self-dual simplex method, Working Paper, Department of Industrial Engineering and Operations Research, University of California at Berkeley.

Adler, I., and N. Megiddo (1985), A simplex algorithm whose average number of steps is bounded between two quadratic functions of the smaller dimension , *J. Assoc. Comput. Mach.* **32**, 871-895.

Adler, I., and R. Shamir (1989), A randomization scheme for speeding up algorithms for linear and convex programming problems with high constraints-to-variables ratio, Technical Report 89-7, DIMACS Center, Rutgers University. To appear in *Math. Programming*.

Adler, I., R.M. Karp, and R. Shamir (1986), A family of simplex variants solving an $m \times d$ linear program in expected number of pivots depending on d only, *Math. Oper. Res.* **11**, 570-590.

Adler, I, R.M. Karp, and R. Shamir (1987), A simplex variant solving an $m \times d$ linear program in $O(\min(m^2, d^2))$ expected number of pivot steps, *J. Complex*. **3**, 371-387.

Alon, N., and N. Megiddo (1990), Parallel linear programming in fixed dimension almost surely in constant time, in *Proc. 31st IEEE Symposium on the Foundations of Computer Science*, IEEE Computer Society Press, Los Alamitos, Calif., 574-582. To appear in *J. Assoc. Comput. Mach* .

Blair, C. (1986), Random linear programs with many variables and few constraints, *Math. Programming* **34**, 62-71.

Borgwardt, K.H. (1977a), *Untersuchungen zur Asymptotik der mittleren Schrittzahl von Simplexverfahren in der Linearen Optimierung*, Dissertation, Kaiserslautern University, Kaiserslautern, Germany.

Borgwardt, K.H. (1977b), Untersuchungen zur Asymptotik der mittleren Schrittzahl yon Simplexverfahren in der Linearen Optimierung, *Methods Oper. Res*. **28**, 332-345.

Borgwardt, K.H. (1978), Zum Rechenaufwand yon Simplexverfahren, *Methods Oper. Res*. **31**, 83-97.

Borgwardt, K.H. (1981), The expected number of pivot steps required by a certain variant of the simplex method is polynomial, *Methods Oper. Res*. **43**, 35-41.

Borgwardt, K.H. (1982), The average number of pivot steps required by the simplex-method is polynomial, *Z. Oper. Res*. **26**, 157-177.

Borgwardt, K.H. (1987), *The Simplex Method: A Probabilistic Approach* , Springer-Verlag, Berlin.

Borgwardt, K.H. (1990), Probabilistic analysis of the simplex method, in *Mathematical Developments Arising From Linear Programming*, J.C. Lagarias and M.J. Todd, eds., Contemporary Mathematics, no. 114, American Mathematical Society, Providence, R.I., 21-34.

Borgwardt, K.H., R. Datum, R. Donig, and G. Joas (1990), Empirical studies on the average efficiency of Simplex variants under rotation symmetry, Report, Institut für Mathematik, Universtät Augsburg, Germany.

Clarkson, K.L. (1991), A Las Vegas algorithm for linear programming when the dimension is small, draft manuscript, AT&T Bell Laboratories, November. (Earlier version with same title in *Proc. 20th Annual Symposium on Theory of Computing* (1988), ACM Press, New York, 452-456.)

Dantzig, G.B. (1963), *Linear Programming and Extensions*, Princeton University Press, Princeton, N.J.

Dyer, M.E., and A.M. Frieze (1989), A randomized algorithm for fixed-dimensional linear programming, *Math. Programming* **44**, 202-212.

Eaves, C., and H. Scarf (1976), The solution of systems of piecewise linear equations, *Math. Oper. Res.* 1, 1-27.

Gass, S.I., and E.L. Salty (1955), The computational algorithm for the parametric objective function, *Naval Res. Logistic Quart.* **2**, 39-45.

Grünbaum, B. (1967), *Convex Polytopes*, John Wiley & Sons, New York.

Haimovich, M. (1983), The simplex algorithm is very good!—On the expected number of pivot steps and related properties of random linear programs, draft, Columbia University, New York.

Kalai, G. (1991), The diameter problem for convex polytopes and *f*-vector theory, in *The Victor Klee Festschrift*, P. Gritzman and B. Sturmfels, eds., American Mathematical Society, Providence, R.I., 387-411.

Kalai, G. (1992a), Upper bounds for the diameter of graphs of convex polytopes, *Discrete Comput. Geom.* **7**, to appear.

Kalai, G. (1992b), A subexponential randomized simplex algorithm (extended abstract), in *Proc. 24th Annual A CM Symposium on Theory of Computing*, ACM Press, New York, to appear.

Kalai, G., and D.J. Kleitman (1992), A quasi-polynomial bound for diameter of graphs of polyhedra, *Bull Amer. Math. Soc.* **24**, in press.

Karmarkar, N. (1984), A new polynomial-time algorithm for linear programming, *Combinatorica* **4**, 373-395.

Klee, V., and P. Kleinschmidt (1987), The d-steps conjecture and its relatives, *Math. Oper. Res.* **12**, 718-755.

Lovász, L. (1980), A new linear programming algorithm—better or worse than the simplex method?, *Math. Intell.* **2**, 141-146.

Matoušek, J., M. Sharir, and E. Welzl (1992), A subexponential bound for linear programming, in *Proc. 8th Annual Symposium on Computational Geometry*, ACM Press, New York, to appear.

Megiddo, N. (1984), Linear programming in linear time when the dimension is fixed, *J. Assoc. Comput. Mach.* **31**, 114-127.

Megiddo, N. (1985), A note on the generality of the self-dual algorithm with various starting points, *Methods Oper. Res.* **49**, 271-275.

Megiddo, N. (1986a), Improved asymptotic analysis of the average number of steps performed by the self dual simplex algorithm, *Math. Programming* **35**, 140-172.

Megiddo, N. (1986b), On the expected number of linear complementarity cones intersected by random and semi-random rays, *Math. Programming* **35**, 225-235.

Megiddo, N. (1987), On the complexity of linear programming, in *Advances in Economic Theory*, T.F. Bewley, ed., Cambridge University Press, New York, 225-268.

Saigal, R. (1983), On some average results for linear complementarity problems, Manuscript, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, Ill.

Seidel, R. (1990), Linear programming and convex hulls made easy, in *Proc. 6th Annual A CM Symposium on Computational Geometry*, ACM Press, New York, 211-215.

Seidel, R. (1991), Low dimensional linear programming and convex hulls made easy, *Discrete Comput. Geom.* **6**, 423-434.

Shamir, R. (1987), The efficiency of the simplex method: A survey, *Manage. Sci.* **33**, 301-334.

Sharir, M., and E. Welzl (1992), A combinatorial bound for linear programming and related problems, in *Proc. 9th Symposium on the Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science no. 577, Springer-Verlag, New York, 569-579.

Smale, S. (1983a), On the average speed of the simplex method of linear programming, *Math. Programming* **27**, 241-262.

Smale, S. (1983b), The problem of the average speed of the simplex method, in *Mathematical Programming: The State of the Art*, A. Bachem, M. Gröt-schel, and B. Korte, eds., Springer-Verlag, Berlin, 530-539.

Todd, M.J. (1986), Polynomial expected behavior of a pivoting algorithm for linear complementary and linear programming problems, *Math. Programming* **35**, 173-192.

Todd, M.J. (1991), Probabilistic models for linear programming, *Math. Oper. Res.* **16**, 671-693.

Welzl, E. (1991), Smallest enclosing disks (balls and ellipsoids), in *New Results and New Trends in Computer Science*, H. Maurer, ed., Lecture Notes in Computer Science no. 555, Springer-Verlag, New York, 359-370.

10. Randomization in Parallel Algorithms

Vijaya Ramachandran[1]
University of Texas at Austin

ABSTRACT

A randomized algorithm is one that uses random numbers or bits during the runtime of the algorithm. Such algorithms, when properly designed, can ensure a correct solution on every input with high probability. For many problems, randomized algorithms have been designed that are simpler or more efficient than the best deterministic algorithms known for the problems. In this chapter, we define a natural randomized parallel complexity class, **RNC**, and give a survey of randomized algorithms for problems in this class.

10.1 INTRODUCTION

In recent years, there has been an explosive growth in the availability and diversity of parallel processors for computation. For the purpose of parallel algorithm design, it is convenient to work with an abstract, simplified machine model, known as the **Parallel Random Access Machine (PRAM)**. The PRAM incorporates the basic elements of a parallel machine and has the property that an algorithm designed for it will perform without significant degradation on most commonly available parallel machines, including

[1] This work was supported in part by NSF grant CCR-89-10707.

shared-memory multiprocessors and fixed interconnection networks. For a survey of PRAM algorithms, both deterministic and randomized, see Karp and Ramachandran (1990).

The PRAM consists of a collection of independent sequential machines, each of which is called a processor, that communicate with one another through a global memory. This is a synchronous model, and a step of a PRAM consists of a read cycle in which each processor can read a global memory cell, a computing cycle in which each processor can perform a unit-time sequential computation, and a write cycle in which each processor can write into a global memory cell. There are many variants of this model, differing in whether a read conflict or a write conflict is allowed and, in the latter case, differing by the method used to resolve a write conflict. Because efficient simulations of the various models are known, we shall not elaborate on these variants.

Parallel algorithms have been designed on the PRAM for a large number of important problems. This has been a rich and exciting area of research in recent years, and many new techniques and paradigms have been developed. However, in spite of impressive gains, some problems have proved to be resistant to attempts to design highly parallel algorithms for their solution. For some other problems, parallel algorithms with good performance have come at the expense of extremely complex and intricate algorithms. In this context, several researchers have turned to randomization in an attempt to obtain better algorithms for these problems.

A randomized parallel algorithm is an algorithm in which each processor has access to a random number generator. The goal is to use this capacity to generate random numbers to come up with an algorithm that solves a problem quickly and with high probability on every input. The requirement that the algorithm achieves its performance guarantee on every input is much stronger than that of demanding good average-case performance and is a highly desirable property in an algorithm that uses randomization. Correspondingly, such algorithms are more difficult to design. Fortunately, there have been a number of successes in the design of such algorithms for parallel machines.

10.2 QUALITY MEASURES FOR RANDOMIZED PARALLEL ALGORITHMS

For the purpose of this exposition, it is convenient to consider a problem as a binary relation $s(x, y)$, where x is the input to the problem and y is a possible output for x. The value of $s(x, y)$ will be true if and only if y is a solution on input *x*. An algorithm for the problem should, on input *x*, either output some *y* such that *s*(*x, y*) is true, or output the fact that no such *y* exists. A Monte Carlo (or one-sided error) randomized algorithm is one that returns a *y* such that *s*(*x, y*) is true with probability at least 1/2 if such a *y* exists; if no such *y* exists, the algorithm always reports failure. Other types of randomized algorithms can be defined, but Monte Carlo algorithms are the ones used most commonly.

Given a Monte Carlo algorithm for a problem, we can improve our confidence in the result supplied by the algorithm from 1/2 to $(1-1/2^k)$ by performing *k* independent runs of the algorithm. Hence, given any $\epsilon > 0$, we can obtain a confidence level greater than $1-\epsilon$ by performing $\lceil \log(1/\epsilon) \rceil$ independent runs of the algorithm.

The notion of a Monte Carlo algorithm applies to algorithms in any model—sequential, parallel, or distributed. In the following sections, we will address the issue of obtaining Monte Carlo algorithms that execute quickly on a PRAM.

10.3 THE RANDOMIZED PARALLEL COMPLEXITY CLASS RNC

In the design and analysis of highly parallel algorithms for various problems, it has been observed that some problems have simple highly parallel algorithms using a relatively small number of processors whereas others have resisted all attempts so far to obtain any algorithm with a significant amount of parallelism. In this context, researchers have identified a natural parallel complexity class **NC**. The class **NC** consists of those problems that have **PRAM** algorithms that run in polylog time, i.e., time polynomial in the logarithm of the input size, with a polynomial number of processors. This class is robust in the sense that the collection of problems it contains does not vary with the machine model used, whether it is a shared-memory machine or a low-diameter interconnection network. It is also the smallest nontrivial class with this property. Although several important problems have been shown to be in **NC**, many others have not. The latter problems are of two types (for our purposes): one type consists of those problems that are either provably not in **NC** by virtue of the fact that they are provably not in **P** (polynomial time), or are highly unlikely to be in **NC** by virtue of the fact that they are complete problems for a larger class (such as **P**); for these problems, we currently know of no technique, randomized or otherwise, to come up with algorithms that run in polylog time with a polynomial number of processors. The other type of problems are those for which no one has come up with either an algorithm to place the problem in **NC** or a completeness result to make it highly unlikely to be placed in **NC**. This latter type includes several important problems such as finding a maximum matching or a depth-first search tree (an important type of spanning tree) in a graph. For these problems, randomization has proved to be a valuable tool in coming up with fast parallel algorithms. For more on parallel complexity classes, see Cook (1981, 1985) and Karp and Ramachandran (1990).

The class **RNC** is the class of problems that can be solved by a Monte Carlo randomized algorithm that runs in polylog time with a polynomial number of processors. Thus, **RNC** is the randomized counterpart of **NC**.

10.4 SOME IMPORTANT PROBLEMS IN RNC

10.4.1 Testing if a Multivariate Polynomial is not Identically Zero

Let Q (x_0, . . ., x_{n-1}) be a multivariate polynomial over a field. Consider the problem of determining if this polynomial is *not* identically zero. It is not known if this problem can be solved deterministically in polynomial time sequentially. However, if we allow randomization, we can come up with a simple Monte Carlo algorithm for the problem, as shown below.

The randomized algorithm is based on the following lemma, which is fairly straightforward to prove by induction on the number of variables n (note that the base case n = 1 follows from the fact that any single variable polynomial of degree n has at most n zeros).

Lemma 1 *Let* $Q(x_0, \ldots, x_{n-1})$ *be a polynomial of degree d with coefficients over a field F. For any set* $I \subseteq F$, *the number of zeros of Q in* $I^n \leq |I|^{n-1} \cdot d$.

The above lemma gives us the following Monte Carlo algorithm to determine if Q is not identically zero (Schwartz, 1980):

1. Choose any set $J \subseteq F$ containing 2d elements.
2. Pick a random element $e = (e_0, \ldots, e_{n-1}) \in J^n$ and evaluate $Q(e)$.
3. If $Q(e) \neq 0$, then report $Q \not\equiv 0$, else report failure.

The probability of success in the case in which Q is not identically zero can be enhanced to $1\text{-}1/2^k$ by repeating steps 2 and 3 k times. This algorithm is easily implemented on a PRAM with n processors by having the ith processor generate e_i. Then, provided $Q(e)$ can be determined quickly given *e*, we have a fast algorithm to test if $Q(e) \not\equiv 0$. In particular, if evaluating *Q* (*e*) is in **NC**, then determining if $Q(x_0, \ldots, x_{n-1}) \not\equiv 0$ is in **RNC**.

10.4.2 Finding a Maximum Matching in a Graph

A matching in an undirected graph is a subset of edges, no two of which share a vertex. A perfect matching is a matching that contains all vertices of the graph. An **RNC** algorithm for the problem of determining if a graph has a perfect matching can be obtained using a theorem of Tutte (1947) that shows that the determinant of a certain matrix of multivariate polynomials becomes identically zero if and only if the graph does not have a perfect matching. This matrix is easily constructed from a description of the input graph. Because computing the determinant of an integer matrix is in **NC**, the problem of determining the existence of a perfect matching is in **RNC** by virtue of the above result on determining if a multivariate polynomial is not identically zero. This result can be extended to place the problem of determining the cardinality of a matching of maximum size, and that of finding a matching with this cardinality, in **RNC** (see Karp et al., 1986; Mulmuley et al., 1987).

The problem of finding a maximum matching in a graph is an important one and one that has received extensive attention in the context of sequential algorithm design. Although the problem is not known to be in **NC**, the above result allows us to find a maximum matching quickly and with high confidence. **RNC** algorithms for several other problems have been obtained in recent years; e.g., Aggarwal and Anderson (1987), Aggarwal et al. (1989), Babai (1986), Gibbons et al. (1988), Karloff (1986), and Ramachandran (1988).

10.5 RANDOMIZATION LEADS TO SIMPLE PARALLEL ALGORITHMS

In practice, the number of processors available for solving a problem is typically much smaller than the problem size. Although massive parallelism may become a reality in the future, it is still of importance to address the issues that arise when the amount of parallelism available is small in comparison to the size of the input. In such a case the efficiency of a parallel algorithm becomes important. This refers to the speedup provided by the parallel algorithm using a fixed number of processors over the best currently known sequential algorithm. A parallel algorithm is considered efficient if the product of its running time and the number of processors it uses is within a polylog factor of the running time of the current best sequential algorithm; it is considered optimal if this product is within a constant factor of the running time of the best sequential algorithm. A parallel algorithm that is efficient (or optimal) when run using a certain number of processors win remain an efficient (or optimal) parallel algorithm when implemented on a smaller number of processors while running proportionately slower. Hence it is of interest to construct efficient parallel algorithms that run very fast, regardless of the number of processors available.

There are a few problems, such as computing a depth-first or breadth-first search tree in a dense graph or performing Gaussian elimination, for which parallel algorithms with very efficient speedups are known even though **NC** algorithms are either not known or highly inefficient. However, most problems for which efficient parallel algorithms are known are in **NC**. Thus, placing a problem in **NC** is generally a first step to obtaining an efficient parallel algorithm for it.

Randomization has proved useful in the design of efficient and optimal parallel algorithms. It has also been useful in developing algorithms that are *simpler* than the best deterministic parallel algorithm known for the problem. Randomization has been applied to achieve these goals for a wide range of problems in graph theory (e.g., Alon et al., 1986; Gazit, 1986; Gibbons et al., 1988; Karp and Wigderson, 1985; and Luby, 1986), sorting (Hagerup, 1991; Rajasekharan and Reif, 1989; Reif and Valiant, 1987), list processing (Vishkin, 1984), computational geometry (Reif and Sen, 1989), string matching (Karl) and Rabin, 1987), linear algebra (Borodin et al., 1982, 1983), and load balancing (Gil, 1991; Gil et al., 1991). The last is a subroutine used in many efficient parallel algorithms to ensure that all of the processors perform about the same amount of work. We illustrate this simplifying potential of randomization with a problem that has received much attention in this context, the maximal independent set problem (Alon et al., 1986; Karp and Wigderson, 1985; Luby, 1986).

The problem of finding a maximal independent set in a graph has been studied extensively in the context of parallel algorithm design. One reason for this is the fact that this problem arises quite frequently in the design of parallel graph algorithms. Another reason is that this problem is a very easy one to solve by a sequential algorithm, but designing a good parallel algorithm for it places quite a few challenges.

Given an undirected graph, a set of vertices is independent if no pair is connected by an edge. The **maximal independent set problem** is the problem of finding an independent set in an input graph with the property that the set cannot be enlarged into a larger independent set that contains it. This problem has a simple sequential algorithm: Fix an ordering of the vertices and examine them in order, adding the examined vertex to the independent set if it contains no edge to a vertex in the set, and discarding the vertex otherwise. Unfortunately, this sequential algorithm does not seem to lend itself to parallelization. Although fast, efficient deterministic parallel algorithms have been developed for this problem, these algorithms tend to be fairly complicated. On the other hand, the following simple randomized algorithm gives a fast parallel algorithm for the problem and is more efficient than any of the deterministic **NC** algorithms known for it.

The randomized algorithm examines each vertex v independently in parallel and assigns the vertex to the independent set with probability $1/(2\delta(v))$, where $\delta(v)$ is the degree of v in the graph. Because this assignment is made simultaneously over all vertices, it is possible that the set constructed is not independent. The algorithm corrects this by examining each edge in the graph, and if the edge has both endpoints in the set, it throws out the vertex with smaller degree (breaking ties randomly). The resulting-set is clearly independent. The algorithm then deletes all neighbors of all vertices in the set from the graph and repeats the procedure with the new, smaller graph. It can be shown that the expected number of edges in the new graph is no more than 7/8 the original number of edges. Hence a logarithmic number of stages of this algorithm suffices to construct an independent set that is maximal in the original graph. This gives an efficient randomized algorithm that runs in $O(\log^2 n)$ time using a linear number of processors.

10.6 ELIMINATING RANDOMIZATION

In some applications, it is desirable or necessary to have a deterministic algorithm. It turns out that some randomized algorithms can be derandomized provided only a limited amount of independence is required. This strategy is applicable to the algorithm described above for the maximal independent set problem. An analysis of that algorithm shows that only pairwise independence is required in the random trials. Hence the algorithm can be made deterministic by searching the entire space in parallel using a quadratic number of processors (Luby, 1986).

An alternate strategy that preserves the number of processors is to compute conditional probabilities while fixing one bit of the output at a time. If k-wise independence suffices for the randomized algorithm, for some constant k, then this gives a deterministic strategy that uses the same number of processors but increases the parallel time by a logarithmic factor. This technique has proved to be a powerful one and has resulted in efficient deterministic **NC** algorithms for several problems using the technique of first constructing an efficient randomized parallel algorithm with limited independence and then transforming the randomized algorithm into a deterministic one (Berger and Rompel, 1989; Luby, 1988; Motwani et al., 1989).

10.7 CONCLUSION

This chapter has described some of the most significant applications of randomization in parallel algorithm design. Randomization has led to fast parallel algorithms and to algorithms that are simple and efficient. These parallel algorithms provide correct solutions with high probability on every input. For most applications, such a performance is almost as satisfactory as a foolproof guarantee. Hence it is not surprising that randomization is a major trend in parallel algorithm design and applications, and we expect the field of randomized parallel algorithms to continue to be a thriving and productive area of research.

REFERENCES

Aggarwal, A., and R.J. Anderson (1987), A random NC algorithm for depth first search, in *Proc. 19th Annual A CM Symposium on Theory of Computing*, ACM Press, New York, 325-334.

Aggarwal, A., R.J. Anderson, and M. Kao (1989), Parallel depth-first search in general directed graphs, in *Proc. 21st Annual ACM Symposium on Theory of Computing*, ACM Press, New York, 297-308.

Alon, N., L. Babai, and A. Itai (1986), A fast and simple randomized parallel algorithm for the maximal independent set problem, *J. Algorithms* **7**, 567-583.

Babai, L. (1986), A Las Vegas-NC algorithm for isomorphism of graphs with bounded multiplicity of eigenvalues, in *Proc. 27th Annual IEEE Symposium on Foundations of Computer Science*, IEEE Computer Society Press, Los Alamitos, Calif., 303-312.

Berger, B., and J. Rompel (1989), Simulating *logsupcn*-wise independence in NC, in *Proc. 30th Annual IEEE Symposium on Foundations of Computer Science*, IEEE Computer Society Press, Los Alamitos, Calif., 2-7.

Borodin, A., S.A. Cook, and N. Pippenger (1983), Parallel computation for well-endowed rings and space-bounded probabilistic machines, *Inform. Contr.* **58**, 113-136.

Borodin, A., J. von zur Gathen, and J.E. Hopcroft (1982), Fast parallel matrix and GCD computations, in *Proc. 23d Annual IEEE Symposium on Foundations of Computer Science*, IEEE Computer Society Press, Los Alamitos, Calif., 65-71.

Cook, S.A. (1981), Towards a complexity theory of synchronous parallel computation, *Enseign. Math*. **27**, 99-124.

Cook, S.A. (1985), A taxonomy of problems with fast parallel algorithms, *Inform. Contr.* **64**, 2-22.

Gazit, H. (1986), An optimal randomized parallel algorithm for finding connected components in a graph, in *Proc. 27th Annual IEEE Symposium on Foundations of Computer Science*, IEEE Computer Society Press, Los Alamitos, Calif., 492-501.

Gibbons, P., R.M. Karp, G. Miller, and D. Soroker (1988), Subtree isomorphism is in Random NC, in *Proc. 3rd Aegean Workshop on Computing* , Springer-Verlag, New York, 43-52.

Gil, J. (1991), Fast load balancing on a PRAM, in *Proc. 3rd IEEE Symposium on Parallel and Distributed Processing*, IEEE Computer Society Press, Los Alamitos, Calif., 10-17.

Gil, J., Y. Matias, and U. Vishkin (1991), Towards a theory of nearly constant time parallel algorithms, in *Proc. 32nd Annual IEEE Symposium on Foundations of Computer Science*, IEEE Computer Society Press, Los Alamitos, Calif., 698-710.

Hagerup, T. (1991), Constant-time parallel integer sorting, in *Proc. 23rd Annual ACM Symposium on Theory of Computing*, ACM Press, New York, 299-306.

Karloff, H.J. (1986), A Las Vegas RNC algorithm for maximum matching, *Combinatorica* **6**, 387-392.

Karp, R.M., and M.O. Rabin (1987), Efficient randomized pattern-matching algorithms, *IBM J. Res. Develop*. **31**, 249-260.

Karp R.M., and V. Ramachandran (1990), Parallel algorithms for shared-memory machines, in *Handbook of Theoretical Computer Science*, vol. A, J. van Leeuwen, ed., Elsevier, New York, 869-941.

Karp, R.M., and A. Wigderson (1985), A fast parallel algorithm for the maximal independent set problem, *J. Assoc. Comput. Mach*. **32**, 762-773.

Karp, R.M., E. Upfal, and A. Wigderson (1986), Constructing a perfect matching is in random NC, *Combinatorica* **6**, 35-48.

Luby, M. (1986), A simple parallel algorithm for the maximal independent set problem, *SIAM J. Comput*. **15**, 1036-1053.

Luby, M. (1988), Removing randomness in parallel computation without a processor penalty, in *Proc. 29th Annual IEEE Symposium on Foundations of Computer Science*, IEEE Computer Society Press, Los Alamitos, Calif., 162-173.

Motwani, R., J. Naor, and M. Naor (1989), The probabilistic method yields deterministic parallel algorithms, in *Proc. 30th Annual IEEE Symposium on Foundations of Computer Science*, IEEE Computer Society Press, Los Alamitos, Calif., 8-13.

Mulmuley, K., U.V. Vazirani, and V.V. Vazirani (1987), Matching is as easy as matrix inversion , in *Proc. 19th Annual ACM Symposium on Theory of Computing*, ACM Press, New York, 345-354.

Rajasekharan, S., and J.H. Reif (1989), Optimal and sublogarithmic time randomized parallel algorithms, *SIAM J. Comput*. **18**, 594-607.

Ramachandran, V. (1988), Fast parallel algorithms for reducible flow graphs, in *Concurrent Computations: Algorithms, Architecture, and Technology*, S.K. Tewksbury, B.W. Dickinson, and S.C. Schwartz, eds., Plenum Press, New York, 117-138.

Reif, J., and S. Sen (1989), Polling: A new randomized sampling technique for computational geometry, in *Proc. 21st Annual ACM Symposium on Theory of Computing*, ACM Press, New York, 394-404.

Reif, J.H., and L.G. Valiant (1987), A logarithmic time sort for linear size networks, *J. Assoc. Comput. Sci*. **34**, 60-76.

Schwartz, J.T. (1980), Fast probabilistic algorithms for verification of polynomial identities, *J. Assoc. Comput. Sci*. **27**, 701-717.

Tutte, W.T. (1947), The factorization of linear graphs, *J. London Math. Soc*. **22**, 107-111.

Vishkin, U. (1984), Randomized speed-ups in parallel computation, in *Proc. 16th Annual ACM Symposium on Theory of Computing*, ACM Press, New York, 230-239.

11. Randomly Wired Multistage Networks

Bruce M. Maggs
NEC Research Institute

ABSTRACT

Randomly wired multistage networks have recently been shown to outperform traditional multistage networks in three respects. First, they have fast deterministic packet-switching and circuit-switching algorithms for routing permutations. Second, they are nonblocking, and there are on-line algorithms for establishing new connections in them, even if many requests for connections axe made simultaneously. Finally, and perhaps most importantly, they are highly fault tolerant.

11.1 INTRODUCTION

Networks derived from hypercubes form the architectural basis of most parallel computers, including machines such as the BBN Butterfly, the Connection Machine, the IBM RP3 and GF11, the iPSC, and the NCUBE. The butterfly, in particular, is quite popular, and has been demonstrated to perform reasonably well in practice. An example of an N-input butterfly ($N = 8$) with depth log $N = 3$ is shown in Figure 11.1. The nodes in this graph represent switches, and the edges represent wires. Messages are typically sent from the switches on level 0, called the **inputs**, to those on level log N, called the **outputs**.

The message-routing algorithm for a butterfly is quite simple. Each message simply follows the unique path of length log N from its source input to its destination output. One problem with this algorithm (and hence the network) is that if some switch or edge along the unique path from input *i* to output *j* (say) becomes congested or fails, then communication between input *i* and output *j* will be disrupted.

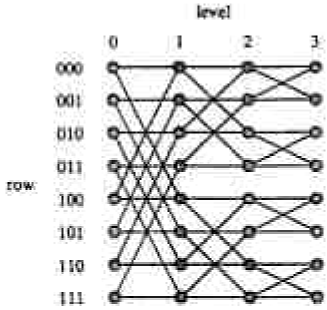

FIGURE 11.1: An 8-input butterfly network. Reprinted, with permission, from Arora et al. (1990). Copyright © 1990 by ACM Press.

11.1.1 Dilated Butterflies

Because message congestion is a common occurrence in real networks, the wires in butterfly networks are typically **dilated**, so that each wire is replaced by a **channel** consisting of two or more wires. In a *d*-dilated butterfly, each channel consists of *d* wires. Because it is harder to congest a channel than it is to congest a single wire in a butterfly, dilated butterflies are better routing networks than simple butterflies (e.g., Koch, 1988; Kruskal and Snir, 1983; Rettberg et al., 1990).

11.1.2 Delta Networks

Butterfly and dilated butterfly networks belong to a larger class of networks called **delta** networks (see, e.g., Kruskal and Snir, 1986). The switches on each level of a delta network can be partitioned into **blocks**. All of the switches on level 0 belong to the same block. On level 1, there axe two blocks, one consisting of the switches that are in the upper *N*/2 rows and the other consisting of the switches that are in the lower *N*/2 rows. In general, the switches in a block *B* of size *M* on level *l* have neighbors in two blocks, B_u, and B_l, on level $l + 1$. The upper block, B_u, contains the switches on level $l + 1$ that are in the same rows as the upper *M*/2 switches of *B*. The lower block, B_l, consists of the switches that are in the same rows as the lower *M*/2 switches of *B*. The edges from *B* to B_u are called the "up-edges," and those from *B* to B_l are called the "down-edges." The three blocks, *B*, B_u, and B_l, and the edges between them are collectively called a **splitter**. The switches in *B* are called the splitter inputs, and those in B_u and B_l are called the splitter outputs.

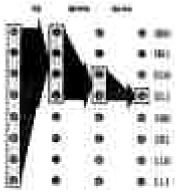

FIGURE 11.2: The logical path from any input to output 011. Reprinted, with permission, from Lisinski et al. (1990). Copyright © 1990 by IEEE.

In a delta network, each input and output are connected by a single logical (up-down) path through the blocks of the network. For example, Figure 11.2 shows the logical path from any input to output 011. In a butterfly, this logical path specifies a unique path through the network, since only one up-edge and one down-edge emanate from each switch. In general, however, each switch may have several up-and down-edges, say *d* of each, and each step of the logical path can be taken on any one of *d* edges.

11.1.3 Multibutterflies

A *d*-dilated butterfly can be thought of as *d* butterflies that are merged together by merging switches that have the same row and level numbers. In a recent paper, Upfal (1989) proposed a more general way to merge butterfly networks. The idea is to permute the order of the switches within each block before merging the networks. Thus, given two *N*-input butterflies G_1 and G_2 and a collection of permutations $\Pi = <\pi_0, \pi_1, \ldots, \pi\log N>$ where $\pi_l : [0, \frac{N}{2^l} - 1] \rightarrow [0, \frac{N}{2^l} - 1]$, a 2-butterfly is formed by first permuting all of the switches in each block on level *l* according to π_l, for $0 \leq l \leq \log N$, and then merging switches with the same row and level numbers. The result, as shown in Figure 11.3, is an N-input graph with depth log N in which each switch has four input edges and four output edges. Of the four output edges at a switch, two are up-edges and two are down-edges (with one up-edge and one down-edge coming from each butterfly). Multibutterflies (i.e., d-butterflies) are composed from d butterflies in a similar fashion using d -1 sets of permutations, $\Pi^{(1)}$,..., Π(d-1), resulting in a depth log N network with $2d \times 2d$ switches.

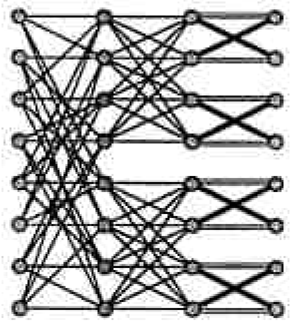

FIGURE 11.3: An 8-input 2-butterfly network. Reprinted, with permission, from Leighton and Maggs (1989). Copyright © 1989 by IEEE.

11.1.4 Expansion

Dilated butterflies have remained the network of choice for many parallel machines. Recent work, however, suggests that this may be about to change. In fact, it now appears as though randomly wired multibutterfly networks (i.e., multibutterfly networks where the permutations $\Pi^{(1)}$,..., Π(d-1) are chosen at random) are superior to dilated butterflies for many message-routing applications. The crucial property that these networks possess is known as **expansion** . In particular, an M-input splitter is said to have (α, β)-expansion if any set of $k \leq \alpha M$ inputs is connected to at least βk up-outputs and βk down-outputs, where $\beta > 1$, ab < 1/2, and α and β are fixed constants. Figure 11.4 shows a splitter with expansion (α, β). Splitters with expansion $\beta > 1$ are known to exist for any $d \geq 3$, and they can be constructed deterministically in polynomial time (e.g., Kahale, 1991; Lubotzky et al., 1988; Upfal, 1989), but randomized wirings typically provide the best possible expansion. In fact, the expansion β of a randomly wired splitter will be close to d-1 with probability close to 1, provided that α is a sufficiently small constant. (For a discussion of the trade-offs between α and β in randomly wired splitters, see Leighton et al. (1991) and Upfal (1989).) Furthermore, the constructions in §§11.3 through 11.5 require $\beta > d/2$, but, at present, there are no known deterministic algorithms for constructing splitters with this much expansion in polynomial time.

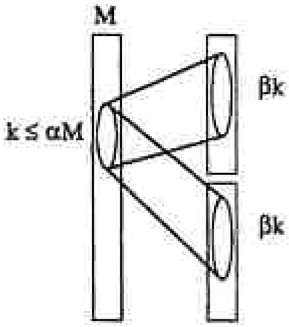

FIGURE 11.4: An *M*-input splitter with expansion property (α, β). Reprinted, with permission, from Arora et al. (1990). Copyright © 1990 by ACM Press.

Splitters with expansion are good for routing because one must block β*k* splitter outputs in order to block *k* splitter inputs. In classical networks such as the butterfly, the reverse is true: it is possible to block 2*k* inputs by blocking only *k* outputs. When this effect is compounded over several levels, the effect is dramatic. In classical networks such as the butterfly, a single fault can block 2^l switches *l* levels back, whereas in a multibutterfly, it takes β^l faults to block a single switch *l* levels back. Splitters with expansion can be constructed deterministically, but for $d \geq 3$, randomized wirings typically provide the best possible expansion.

11.1.5 History

Randomly wired multibutterfly networks have been discovered several times. Bassalygo and Pinsker (1974) used random multibutterfly-like networks to construct the first nonblocking network with size of $O(N\log N)$ and depth of $O(\log N)$. Fahlman (1980) proposed a related randomly wired network called the Hashnet. More recently, Upfal (1989) coined the term *multibutterfly* and provided a simple deterministic algorithm for routing any permutation of *N* messages in $O(\log N)$ steps on an *N*-input multibutterfly network.

(In fact, Upfal's algorithm can be pipelined to route log *N* permutations in *O*(log *N*) steps.) Leighton and Maggs (1989) proved that a randomly wired multibutterfly is an efficient routing network even when many of the switches are faulty. Later, Arora et al. (1990) developed a circuit-switching algorithm for the multi-Benes network and showed that it can be used to establish connections in a nonblocking fashion. Most recently, DeHon et al. (1991) designed a 64-processor switching network using a randomly wired delta network for processor-to-memory communications.

11.1.6 Outline

The remainder of this chapter is organized as follows. Section 11.2 describes Upfal's algorithm for packet switching on multibutterfly networks. Next, §11.3 presents a multibutterfly algorithm for circuit switching. Section 11.4 describes a strategy for tolerating faults. Finally, §11.5 sketches an algorithm for establishing connections in a randomly wired nonblocking network.

11.2 PACKET SWITCHING

In a one-to-one packet-routing problem, each input sends a packet to a distinct output. The goal of the routing algorithm is to deliver the packets to their destinations as quickly as possible, subject to the constraint that at each time step each edge can transmit at most one packet. There may also be restrictions on the number of packets that can be queued at any one switch.

Upfal (1989) proved that an *N*-input *d*-butterfly with expansion (α, β) can solve any one-to-one packet-routing problem in *O*(log *N*) steps using a simple greedy algorithm. Moreover, he showed that by using pipelining, *O*(log *N*) problems can also be routed in *O*(log *N*) steps. The result is important because the only other known deterministic on-line linear-hardware *O*(log *N*)-step packet-routing algorithm (Leighton, 1985) requires the use of the AKS sorting circuit (Ajtai et al., 1983), which is more complicated and has larger constant factors.

Upfal's algorithm starts by partitioning the packets into "waves" so that at most one packet in each wave is destined for any set of *L* contiguous outputs. One way to do this is to group packets into the same wave if they are in the same permutation and their destinations are congruent modulo *L*. If there are *P* permutations to be routed, this results in the formation of at most *PL* waves. In general, we will set $L = 1/(2\alpha)$ because then we win be guaranteed that at most $M/(2L) = \alpha M$ packets in any wave will ever pass through the up-(or down-) edges of any *M*-input splitter of the multibutterfly (for any *M*). This will allow us to apply the (α, β) expansion property to the set of inputs of any splitter occupied by the packets of a single wave at any time. (E.g., if *k* inputs of a splitter contain packets of a single wave that want to traverse up-edges, then these inputs are connected to at least β*k* up-outputs.) This is because packets going through the *M*/2 up (or *M*/2 clown) splitter outputs can only be destined for the descendant set of *M*/2 contiguous multibutterfly outputs.

The routing of the packets proceeds in stages, each stage consisting of an even and odd phase, and each phase consisting of 2*d* steps. In even phases, packets axe sent from even levels to the next (odd) level, and in odd phases, packets axe sent from the odd levels to the next (even) level. The edges connecting levels axe colored in 2*d* colors so that each node is incident to one edge of each color. In each phase, we process the colors in sequence, one step per color. For each color, we move a packet forward along an edge with that color if there is a packet in the switch at the tail of the edge that wants to go in that direction (up or down) and if there is no packet in the switch at the head of the edge. Alternatively, if there is a packet in the switch at the head of the edge and if it is in a later wave than the packet at the tail of the edge, then the two packets axe swapped, so that the packet in the earlier wave moves forward. Note that every switch processes and/or contains at most one packet at any step.

The following theorem summarizes the performance of Upfal's algorithm.

Theorem 11.1 (Leighton and Maggs, 1989; Upfal, 1989) *On an N-input multibutterfly with expansion* (α, β), *Upfal's algorithm routes P permutations in* $O(P + \log N)$ *steps*.

11.3 CIRCUIT SWITCHING

In a one-to-one circuit-switching problem, each input wishes to establish a connection (path) to a distinct output. The connections must not intersect at any switch or edge. The goal of the circuit-switching algorithm is to find the connections as quickly as possible.

Arora et al. (1990) present an $O(\log N)$-bit-step algorithm for circuit switching on multibutterfly networks. The only previously known $O(\log N)$-bit-step algorithms for circuit switching relied on the AKS sorting circuit (Ajtai et al., 1983)or used randomness on the hypercube (Aiello et al., 1990). (Recently, Leighton and Plaxton (1990) have developed an $O(\log N)$-bit-step randomized sorting algorithm for the butterfly.)

11.3.1 Unique Neighbors

The circuit-switching algorithm requires the splitters in the multibutterfly to have a special unique neighbor property. An *M*-input splitter is said to have the (α, δ) **unique neighbor property** if in every subset *X* of $k \leq \alpha M$ inputs, there are δ*k* nodes in *X* that have an up-output neighbor that is not adjacent to any other node in *X*, and there axe δ*k* nodes in *X* that have a down-output neighbor that is not adjacent to any other node in *X*. It is relatively easy to prove (see Arora et al., 1990) that any splitter with (α,β) expansion has the (α, δ) unique neighbor property where δ = 2β/*d* -1, provided that α > *d*/2. Randomly wired multibutterflies are known to have expansion (α, β) where β *d*/2 (Leighton and Maggs, 1989; Upfal, 1989). Explicit constructions of such splitters are not known, however.

11.3.2 The Algorithm

In order for the algorithm to succeed, the number of paths passing through each *M*-input splitter must be at most α*M*. Thus, in an *N*-input network, we make connections between the *N/L* inputs and outputs only in rows that are multiples of *L*, where *L* is some fixed constant greater than 1/α.

There is a simple algorithm for extending paths from one level to the next in an *M*-input splitter with the (α, δ) unique neighbor property. The basic idea is that those paths at switches with unique neighbors can be extended without worrying about blocking any of the other paths. Paths are extended by repeating steps of the following type. First, every unextended path sends out a proposal to his neighbors among the splitter outputs in the desired direction (up or down). Next, every output that receives precisely one proposal sends back its acceptance to that proposal. Finally, every unextended path that receives an acceptance advances to one of its accepting outputs. In each step, the fraction of unextended paths drops by a factor of (1 -δ). Thus, after *O*(log *M*) phases, all of the paths are extended. By applying this algorithm one level at a time, it is possible to establish paths from the inputs to the outputs of an *N*-input multibutterfly with the (α, δ) unique neighbor property in *O* (log^2 *N*) bit-steps.

A more sophisticated algorithm is needed to construct the paths in *O*(log *N*) bit-steps. Given a set of paths that need to be extended at an *M*-input splitter, the algorithm does not wait *O*(log *M*) time for every path to be extended before it begins the extension at the next level. Instead, it executes path extension steps until the number of unextended paths falls to some fraction ρ of its original value, where ρ is a fixed constant that depends on *d*. Then the path extension process can start at the next level.

The danger is that the paths left behind may find themselves blocked by the time they reach the next level. To ensure that this does not happen, stalled paths send out "placeholders" to all of their neighbors at the next level, and henceforth the neighbors with placeholders participate in path extension at the next level, as if they were paths. Of course, the neighbors holding placeholders must in general extend in both the upper and the lower output portions of the splitter, because they do not yet know which path will ultimately use them. Notice that a placeholder not only reserves a spot that may be used by a path at a future time but also helps to chart out the path by continuing to extend ahead.

In order to prevent placeholders from multiplying too rapidly and clogging the system—because if the fraction of inputs of a splitter that are trying to extend rises above α, the path extension algorithm ceases to work—we need to ensure that as stalled paths get extended, they send cancellation signals to the placeholding nodes ahead of them to tell them they are not needed anymore. When a placeholding node gets cancellations from all the nodes that had requested it to hold their place, it ceases its attempts to extend. It also sends cancellations to any nodes ahead of it that may be holding a place for it.

The $O(\log N)$-bit-step algorithm alternates between two types of phases. First, path extension steps are executed until the fraction of unextended paths in each splitter drops by a factor of ρ. In this phase, each path is restricted to extending forward by at most one level. We refer to the first wave of paths and placeholders to arrive at a level as the wavefront. The wavefront moves forward by one level during each phase. If a path or placeholder in the wavefront is not extended, then at the end of the phase it sends placeholders to all of its neighbors. In the second phase, cancellations are passed through the network. They travel a distance of C, where C is some fixed constant that depends on ρ and d.

The performance of the circuit-switching algorithm is summarized in the theorem below.

Theorem 11.2 (Arora et al., 1990) *On an N-input multibutterfly with expansion* (α, β), $\beta\ d/2$, *the algorithm solves any one-to-one circuit-switching problem in* $O(\log N)$ *bit-steps*.

11.4 FAULT TOLERANCE

Leighton and Maggs (1989) showed that multibutterfly networks are highly fault tolerant. In particular, they proved that no matter how an adversary chooses *k* switches to fail, there will be at least *N*-*O*(*k*) inputs and *N*-*O*(*k*) outputs between which permutations can be routed in *O*(log *N*) steps. Note that this is the best that could be hoped for in general, because the adversary can choose to make Ω(*k*) inputs and Ω(*k*) outputs faulty. Thus, the multibutterfly is the first bounded-degree network known to be able to sustain large numbers of faults with minimal degradation in performance.

The strategy for tolerating faults consists of two stages: erasure of outputs and fault propagation.

Each splitter in the multibutterfly is examined in the erasure stage. If more than an *ε* fraction of the splitter inputs are faulty, where *ε* = 2α(β'-1) and $\beta' = \beta - \lfloor \frac{d}{2} \rfloor$, then the splitter, as well as all descendant switches and outputs, is erased from the network. The erasure of an *M*-input splitter causes the removal of M multibutterfly outputs and accounts for at least *εM* faults. Hence, at most *k*/*ε* = (*kL*) / (β'-1) = *O*(*k*) multibutterfly outputs are removed by this process.

Next, working from level log *N* back to level 0, each switch is examined in the fault propagation stage to see if at least half of its upper outputs lead to faulty switches that have not been erased or if at least half of its lower outputs lead to faulty switches that have not been erased. If so, then the switch is declared faulty (but not erased). It is not difficult to prove (see Leighton and Maggs, 1989) that at most *k* / (α'-1) additional switches are declared faulty at each level by this process. Hence, at most *O*(*k*) multibutterfly inputs will be faulty (declared or otherwise).

We now erase all the remaining faulty switches. This leaves a network with *N*-*O*(*k*) inputs and *N*-*O*(*k*) outputs. Moreover, every input in every splitter is linked to $\lceil \frac{d}{2} \rceil$ functioning upper outputs (if the descendant multibutterfly outputs exist) and $\lceil \frac{d}{2} \rceil$ functioning lower outputs (if the corresponding multibutterfly outputs exist). Hence every splitter has an (α,β') expansion property. Thus, we can apply theorems 11.1 and 11.2 with β replaced by β' to show that the network can solve packet-switching and circuit-switching problems on the working inputs and outputs in *O*(log *N*) steps.

11.5 NONBLOCKING NETWORKS

In a **nonblocking network**, inputs are connected to outputs with node-disjoint paths, as they were in §11.3. The inputs, however, are not all required to make their requests for connections at the same time. Inputs may wait to make their requests and may later break connections and request new ones. The main invariant obeyed by a nonblocking network is that any unused input-output pair can be connected by a path through unused switches, no matter what paths have previously been established. The 6-terminal graph shown in Figure 11.5 is an example of a nonblocking network. In particular, if Bob is talking to Alice and Ted is talking to Carol, then Pat can still call Vanna.

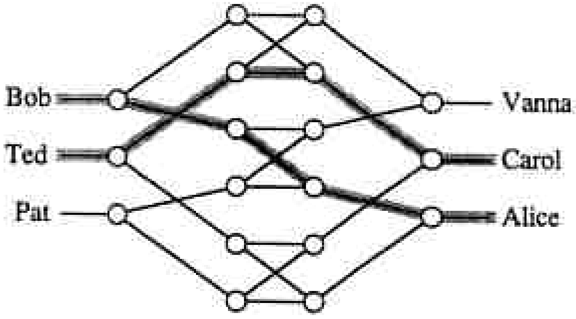

FIGURE 11.5: A nonblocking network with 3 inputs and 3 outputs. Reprinted, with permission, from Arora et al. (1990). Copyright © 1990 by ACM Press.

The existence of a bounded-degree strict-sense nonblocking network with size $O(N \log N)$ and depth $O(\log N)$ was first proved by Bassalygo and Pinsker (1974). Unfortunately, there has not been much progress on the problem of setting the switches so as to realize the connection paths since then. Until recently, no algorithm was known that could cope with simultaneous requests for connections in any $O(N \log N)$-size nonblocking network.

Arora et al. (1990) discovered an $O(N \log N)$-switch nonblocking network for which each path connection can be made on-line in $O(\log N)$ bit-steps. The algorithms work even if many calls are made at once—every call still gets through in $O(\log N)$ bit-steps, no matter what calls were made previously and no matter what calls are currently active, provided that no two inputs try to access the same output at the same time.

The nonblocking network is called a **multi-Benes network**. A multi-Benes network is constructed by combining expanders and the Benes network in much the same way that expanders and butterflies are combined to fore a multibutterfly. As shown in Figure 11.6, a multi-Benes network is essentially a reversed multibutterfly followed by a multibutterfly.

As in the circuit-switching algorithm, the network must be lightly loaded by some fixed constant factor L, where $L > 1/\alpha$. Since the other inputs and outputs are not used, the first and last $\log_2 L$. levels of the network can be removed, and the *N/L*, inputs and outputs can each be connected directly to their *L* descendants and ancestors on levels $\log_2 L$ and $2\log_2 N$-$\log_2 L$, respectively.

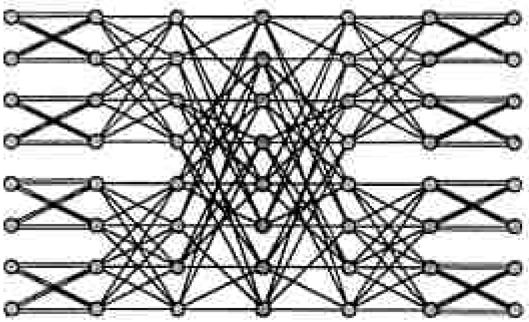

FIGURE 11.6: An 8-input 2-multi-Benes network.

The basic idea is to treat the switches through which paths have already been established as if they were faulty and to apply the fault propagation techniques from §11.4 to the network. In particular, we define a node to be "busy" if there is a path currently routing through it, and we recursively define a node to be "blocked" according to the following rule. Working backwards from level $2\log_2 N$-$\log_2 L$-1 to level $\log_2 N$, a switch is declared blocked if more than 2β-*d*-1 of its up (or down) neighbors on level *l*+1 are busy or blocked. From level $\log_2 N$-1 to level $\log_2 L$, a switch is declared blocked if more than 4β-*d*-2 of its 2*d* neighbors on level *l*+1 are busy or blocked. A switch that is neither busy nor blocked is said to be "working."

Two important properties can be proved about the network switches. First, for β > 2*d*/3+2/3 and *L* > 1/2α (3β-2*d*-2), at most a 2α fraction of the switches in any block are declared to be blocked. Thus, all of the unused inputs are working. As a consequence, no matter what paths have already been established, any unused input can reach any unused output. Second, for β > *d*/2, the network of working switches has a (α, 1/*d*) unique neighbor property. As a consequence, the circuit-switching algorithm from §11.3 can be used to establish new paths, even if many requests for connections are made simultaneously.

REFERENCES

Aiello, W., T. Leighton, B. Maggs, and M. Newman (1990), Fast algorithms for bit-serial routing on a hypercube, in *Proc. 1990 ACM Symposium on Parallel Algorithms and Architectures*, ACM Press, New York, 55-64.

Ajtai, M., J. Komlos, and E. Szemeredi (1983), Sorting in clog *n* parallel steps, *Combinatorica* **3**, 1-19.

Arora, S., T. Leighton, and B. Maggs (1990), On-line algorithms for path selection in a non-blocking network, in *Proc. 22nd Annual ACM Symposium on Theory of Computing*, ACM Press, New York, 149-158.

Bassalygo, L.A., and M.S. Pinsker (1974), Complexity of optimum non-blocking switching network without reconnections, *Probl. Inform. Transm*. **9**, 64-66.

DeHon, A., T. Knight, and H. Minsky (1991), Fault-tolerant design for multistage routing networks, in *Proc. International Symposium on Shared Memory Multiprocessing*, Information Processing Society of Japan.

Fahlman, S.E. (1980), The Hashnet interconnection scheme, Technical Report CMU-CS-80-125, Department of Computer Science, Carnegie Mellon University, Pittsburgh, Pa.

Kahale, N. (1991), Better expansion for Ramanujan graphs, in *Proc. 32nd Annual Symposium on Foundations of Computer Science*, IEEE Computer Society Press, Los Alamitos, Calif., 398-404.

Koch, R.R. (1988), Increasing the size of a network by a constant factor can increase performance by more than a constant factor, in *Proc. 29th Annual Symposium on Foundations of Computer Science*, IEEE Computer Society Press, Los Alamitos, Calif., 221-230.

Kruskal, C.P., and M. Snir (1983), The performance of multistage interconnection networks for multiprocessors, *IEEE Trans. Comput*. **C-32**, 1091-1098.

Kruskal, C.P., and M. Snir (1986), A unified theory of interconnection network structure, *Theor. Comput. Sci.* **48**, 75-94.

Leighton, F.T. (1985), Tight bounds on the complexity of parallel sorting, *IEEE Trans. Comput.* **C-34**, 344-354.

Leighton, T., and B. Maggs (1989), Expanders might be practical: Fast algorithms for routing around faults in multibutterflies, in *Proc. 30th Annual Symposium on Foundations of Computer Science*, IEEE Computer Society Press, Los Alamitos, Calif, 384-389.

Leighton, T., and G. Plaxton (1990), A (fairly) simple circuit that (usually) sorts, in *Proc. 31st Annual Symposium on Foundations of Computer Science*, IEEE Computer Society Press, Los Alamitos, Calif., 264-274.

Leighton, T., C.L. Leiserson, and M. Klugerman (1991), Theory of parallel and VLSI computation, Research Seminar Series Report MIT/LCS/RSS 10, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge.

Lisinski, D., T. Leighton, and B. Maggs (1990), Empirical evaluation of randomly-wired multistage networks, in *Proc. 1990 International Conference on Computer Design*, IEEE Computer Society Press, Los Alamitos, Calif., 380-385.

Lubotzky, A., R. Phillips, and P. Sarnak (1988), Ramanujan graphs, *Combinatorica* **8**, 261-277.

Rettberg, R.D., W.R. Crowther, P.P. Carvey, and R.S. Tomlinson (1990), The monarch parallel processor hardware design, *Computer* **23**, 18-30.

Upfal, E. (1989), An $O(\log N)$ deterministic packet routing scheme, in *Proc. 21st Annual ACM Symposium on Theory of Computing*, ACM Press, New York, 241-250.

12.

Missing Pieces, Derandomization, and Concluding Remarks

J. Michael Steele,[1]
University of Pennsylvania

No survey is ever complete, and completeness is especially elusive for a survey of a rapidly evolving subject like the interface of probability and the theory of algorithms. As we go to press, there is late-breaking news of stunning progress on transparent proof techniques, which, in a nutshell, are methods that (in some versions) allow one to test the validity of alleged proofs by applying tests that will fail with probability 1/2 unless the proof is valid. Some of the foundations that underlie this development are touched on in Chapters 4 and 5, but, even so, an honest sketch of the new ideas in transparent proof techniques would require a substantial excursion into the most modern bits of computational complexity theory. Rather than take on that excursion, we have to be content with referring the reader to the journalistic accounts of Kolata (1992) and Cipra (1992) and the more technical discussion of Johnson (1992). The latter contains pointers to the appropriate scientific literature.

If one studies the engineering of these recent advances in transparent proof techniques, one finds many structures that were first supported by probabilistic thinking. As it happens, one also finds that considerable effort was subsequently invested to replace most of the original probabilistic scaffolding by constructions that could be called purely deterministic. To probabilists, this passion for excising randomized constructions seems curious, but many computer scientists and combinatorists feel themselves to be

[1] Research supported in part by NSF grant DMS88,-12868, AFOSR grant 89-0301; ARO grant DAAL03-89-G-0092, and NSA grant MDA-904-H-2034.

on philosophically shaky ground with randomized constructions. Thus, in many contexts, such constructions are accorded only the status of a pure existence proof and, almost always, they are seen as lacking at least some of the pristine virtue of deterministic constructions.

The classical way to atone for the easy virtue of probabilistic constructions has been to supply deterministic replacements, and a famous illustration of this tradition arises in the discussion of expander graphs in Chapter 11. Though it is shockingly easy to show the existence of all sorts of expander graphs via probability, many researchers seemed to breathe a sigh of relief when deterministic constructions for good expanders were developed, even when the new constructions called on methods as sophisticated as those used by Lubotzky et al. (1988).

Although most of the discoveries provoked by the urge to replace a randomized construction with a deterministic one have turned out to be rather specialized, there are recent developments that change this situation and offer enticing suggestions of an important new theory of **derandomization**. Instead of pursuing clever ad hoc constructions, the new idea is to look for algorithmic procedures that replace the very steps employed in the randomized construction. This shift in perspective turns out to be very fruitful, and a good sense of its power can found in Chapter 15 of the important new book *The Probabilistic Method*, by Alon et al. (1992). Further, because of the remarkable utility of the Lovász Local Lemma in the more subtle probabilistic constructions, the recent algorithmization of the Local Lemma by Beck (1992) and Alon (1992) offers a compelling validation of the derandomization concept.

In addition to its useful discussion of derandomization, the volume of Alon et al. (1992) also provides charming introductory treatments of at least two other topics that may seem underrepresented in this survey, **graph algorithms** and **random graphs**. The latter topic is also well covered in the treatise *Random Graphs* by Bollobás (1985), which is a bible for any serious student of random graphs. From the probabilists' point of view, two of the most important recent links to these areas are to percolation theory and to the Chen-Stein method of Poisson approximation. Much of the recent progress in percolation theory is beautifully introduced by Grimmett (1989), and the recent treatise on Poisson approximation of Barbour et al. (1992) surely belongs on the bookshelf of anyone interested in probability, graphs, or discrete algorithms.

Another important sphere of probability in the service of algorithms that some may see as underrepresented here is the **analytical** probabilistic analysis of algorithms. This area can be characterized by the use of explicit combinatorial calculations and generating functions to calculate the means and variances of algorithm running times and other features of interest. Historically, the critical parts of such calculations tend to be more closely connected with real and complex analysis than with probability theory, but the language of probability always drives the problem formation and increasingly contributes to the analysis. Much of this tradition springs from seminal work of Donald Knuth, with many illustrations of the central themes found in his now classical books *The Art of Computer Programming*, vols. 1-3 (Knuth, 1973). A more recent and introductory treatment that is sympathetic in approach is the text of Bender and Williamson (1991), which also can be commended for the insights it offers into asymptotic analyses assisted by generating functions. Another recent volume that anyone involved with the analytical tradition should read is Will (1990), which christens "generatingfunctionology" as a field in itself and also offers up many of the field's secrets in a way in which they can be enjoyably mastered.

A final volume that deserves mention here is the recent collection *Probabilistic Combinatorics and Its Applications*, edited by Bollobás (1991). The seven essays in this collection are all of great interest to the field, and each points toward many lively research topics. In particular, the essay by F.R.K. Chung (1991) provides quite another perspective on derandomization theory and illustrates many of the subtleties that perplex investigators who examine randomness in a quest to find acceptable surrogates.

REFERENCES

Alon, N. (1992), A parallel algorithmic version of the Local Lemma, *Rand. Struct. Algorithms* **2**, 367-377.

Alon, N., J. Spencer, and P. Erdös (1992), *The Probabilistic Method* , Wiley-Interscience Series in Discrete Optimization, John Wiley & Sons, New York.

Barbour, A.D., L. Hoist, and S. Janson (1992), *Poisson Approximation* , Oxford Studies in Probability no. 2, Clarendon Press, Oxford.

Beck, J. (1992), An algorithmic approach to the Lovász Local Lemma I, *Rand. Struct. Algorithms* **2**,343-366.

Bender, E.A., and S.G. Williamson (1991), *Foundations of Applied Combinatorics*, Addison-Wesley, Reading, Mass.

Bollobás, B. (1985), *Random Graphs*, Academic Press, New York.

Bollobás, B., ed. (1991), *Probabilistic Combinatorics and Its Applications* , Proceedings of Symposia in Applied Mathematics no. 44, American Mathematical Society, Providence, R.I.

Chung, F.R.K. (1991), Constructing random-like graphs, in *Probabilistic Combinatorics and Its Applications*, B. Bollobás, ed., American Mathematical Society, Providence, R.I., 21-56.

Cipra, B.A. (1992), Theoretical computer scientists develop transparent proof technique, *SIAM News* **25** (May), 1.

Graham, R.L., D.E. Knuth, and O. Patashnik (1989), *Concrete Mathematics: A Foundation for Computer Science*, Addison-Wesley, Reading, Mass.

Grimmett, G. (1989), *Percolation*, Springer-Verlag, New York.

Johnson, D.S. (1992), The NP-completeness column: An ongoing guide (23rd edition): The tale of the second prover, *J. Algorithms* **13** (to appear in September issue).

Knuth, D.E. (1973), *The Art of Computer Programming*, (vols. 1-3), Addison-Wesley, Reading, Mass.

Kolata, G. (1992), New short cut found for long proofs, *New York Times*, April 6, section C.

Lubotzky, A, R. Phillips, and P. Sarnak (1988), Ramanujan graphs, *Combinatorica* **8**, 261-277.

Wilf, H.S. (1990), *Generatingfunctionology*, Academic Press, Boston.