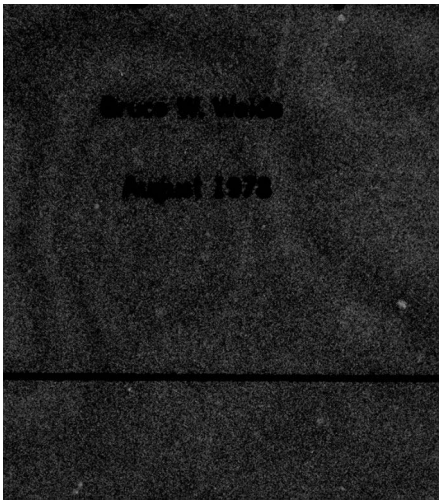
AD-A061 150 CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER --ETC F/G 12/1
STATISTICAL METHODS IN ALGORITHM DESIGN AND ANALYSIS. (U)
AUG 78 B W WEIDE N00014-76-C-0370
UNCLASSIFIED CMU-CS-78-142 NL

1 OF 2
ADA
061150




Bruce W. Webb
August 1978

(14) CMU-CS-78-142

(6) Statistical Methods in Algorithm Design and Analysis.

(9) Doctoral thesis,

(10) Bruce W. Weide

Dept. of Computer Science

Carnegie-Mellon University

Pittsburgh, PA 15213

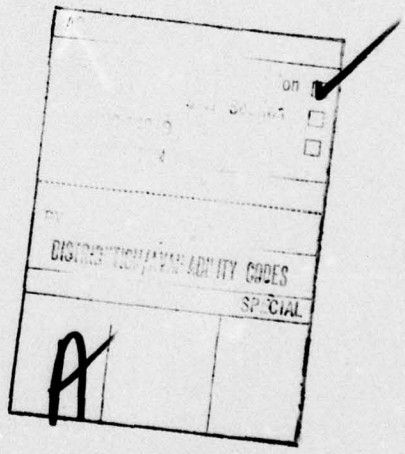
(11) August 1978 (12) 185 p.

Submitted to Carnegie-Mellon University in partial fulfillment of the requirements for the degree of Doctor of Philosophy

(15) This research was supported by the Office of Naval Research under Contract N0014-76-C-0370

78 11 09 017

403081 LB



Copyright © 1978 by the author. All rights reserved. No part of this document may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the author.

Abstract

The use of statistical methods in the design and analysis of discrete algorithms is explored. Among the design tools are randomization, ranking, sampling and subsampling, density estimation, and "cell" or "bucket" techniques. The analysis techniques include those based on the design methods as well as the use of stochastic convergence concepts and order statistics.

The introductory chapter contains a literature survey and background material on probability theory. In Chapter 2, probabilistic approximation algorithms are discussed with the goal of exposing and correcting some oversights in previous work. Some advantages of the proposed solution to the problems encountered are the introduction of a model for dealing with random problems, and a set of methods for analyzing the probabilistic behavior of approximation algorithms which permit consideration of fairly complex algorithms in which there are dependencies among the random variables in question.

Chapter 3 contains many useful design and analysis tools such as those mentioned above, and several examples of the uses of the methods. Algorithms which run in linear expected time for a wide range of probabilistic assumptions about their inputs are given for problems ranging from sorting to finding all nearest neighbors in a point set in k dimensions. Empirical results are presented which indicate that the sorting algorithm, Binsort, is a good alternative to Quicksort under most conditions. There are also new algorithms for some selection and discrete optimization problems.

Finally, Chapter 4 describes the uses of results from order statistics to analyze greedy algorithms and to investigate the behavior of parallel algorithms. Among the results reported here are general theorems regarding the distribution of solution values for optimization problems on weighted graphs. Many recent results in the literature, which apply for certain distributions of edge weights and for specific problems, follow as immediate corollaries from these general theorems.

Contents

Acknowledgements

1. Introduction and Summary

1.1. Previous Work 1-2

1.2. Summary of Chapter 2: Stochastic Convergence, Probabilistic Algorithms 1-5

1.3. Summary of Chapter 3: Randomization and Sampling 1-7

1.4. Summary of Chapter 4: Order Statistics 1-9

1.5. Background Material 1-9

1.5.1. Notation 1-10

1.5.2. Basic Probability Theory 1-11

1.5.3. Random Structures 1-15

1.6. Conclusions and Further Work 1-18

2. Probabilistic Algorithms and Stochastic Convergence

2.1. Stochastic Convergence 2-1

2.2. Random Problem Models 2-12

2.3. History of Confusion 2-19

2.4. Strong Success in the Independent Model 2-24

2.5. Example: The Traveling Salesman Problem 2-25

2.6. Conclusions 2-40

3. Randomization and Sampling

3.1. Classification of Algorithms 3-2

3.2. Classification of Statistical Procedures 3-4

3.3. Design Principles for Randomized Algorithms 3-7

3.3.1. Randomization 3-8

3.3.2. Approximation in Rank 3-11

3.3.3. Estimation and Subsampling 3-15

3.3.4. Empirical Distribution Functions 3-19

3.4. Examples 3-29

3.4.1. Sorting and Searching 3-30

3.4.2. Selection 3-39

3.4.3. Discrete Optimization Problems 3-51

3.4.4. Geometrical Problems 3-58

3.5. Conclusions 3-75

4. Order Statistics

4.1. Expected Values and Asymptotic Distributions 4-1

4.2. Examples 4-6

4.2.1. Greedy Algorithms 4-6

4.2.2. Parallelism 4-18

4.2.3. Problem Decomposition for Multiprocessors 4-23

4.3. Conclusions 4-30

5. References

Acknowledgements

The suggestion that there might be something more to statistics than "mere computation" was made by my thesis advisor, Mike Shamos, about three years ago. He proceeded to explore how the algorithm design tools of computer science could be applied to discrete statistical computations, having written his own Ph.D. thesis on their application to geometrical computations. Meanwhile, I examined the opposite approach: How could statistical tools help in the design and analysis of algorithms for computer science? I am happy to acknowledge that Mike is responsible for my interest in statistics as well as algorithms. It is enough to ask that one's thesis advisor help with technical matters, but for him to translate articles from the Russian originals is more than should be expected. Nevertheless, that is exactly what Mike did. He is also the source of most of the problem ideas and started me thinking about many of the solutions in this thesis, and I am proud to consider him my friend.

Without the patience and assistance of the other members of my thesis committee, however, this work would still be a proposal. Jon Bentley provided many insightful suggestions regarding the algorithmic aspects, and more than he admits regarding the statistical ones. Bill Eddy put up with my constant questions about probability theory and statistics, answering most of them immediately and spending considerable effort leading me in search of answers to the others. Bill offered many

particularly good ideas about the material in Chapter 2, and both he and Jon were always available for consultation about technical and non-technical problems alike. Jay Kadane and H.T. Kung also made many important suggestions, without which I would still be trying to prove some of the theorems of Chapters 2 and 4.

Of course, many other people helped with the technical problems and with my writing style. At the risk of overlooking someone, I would especially like to thank Tom Andrews, Gerard Baudet, Kevin Brown, Peter Denning, Diane Detig, Therese Flaherty, Sam Fuller, Paul Hilfinger, David Jefferson, John Lehoczky, Takao Nishizeki, Larry Rafsky, John Robinson, Jim Saxe, Joe Traub, and Jay Wolf. Also, thanks to Lee Cooprider, Brian Reid, and Mark Sapsford for their assistance with CMU's marvelous document production facilities. In keeping with tradition, I suppose that I should assume responsibility for any remaining errors in the manuscript, which I hereby do. However, I *am confident* that there are not too many left because of the careful perusal of early drafts by several of these people.

Generous financial support during my four years at CMU was provided by the National Science Foundation and by IBM in the form of graduate fellowships, and by my parents in forms too numerous to list here.

Finally, I would like to thank my family, especially my parents, Harley and Betty Jo Weide, and several close friends who helped make this experience a pleasant as well as an educational one. Ann Kolwitz, Jay and Ellen Wolf, Diane Detig, Dave and Moddy McKeown, and Jon and Judy Rosenberg saved me from overwork and boredom on several occasions, as did the members of Turing's Machine, the Arpanets, the Jive Turkeys, and last but certainly not least, SIGLUNCH. To all these people I owe a sincere debt of gratitude for their continuing friendship.

1. Introduction and Summary

Until quite recently, research in the design and analysis of discrete algorithms has been devoted to the "worst-case" question: At worst, how bad is this algorithm? The results produced by this effort are of considerable intrinsic interest, and even of occasional practical value, but the label "pessimist" is often attached to computer scientists who pursue these issues. Investigations of the "typical" behavior of algorithms are actually motivated more by pragmatism than by optimism, but they are frustrated by the difficulty of dealing with general probabilistic models.

The major point of this thesis is that certain parts of probability theory and statistics, which are not really difficult to learn, provide valuable tools for the exploration of some practical issues in algorithm design and analysis.[1] Thus, while many of the results reported here are apparently only of a theoretical nature, others are directly applicable to real-world situations. Although the contributions of this work include these results, they constitute only a minor part of the motivation for it: most are simply demonstrations that results can be produced using the proposed methods.

The most important parts of this thesis are the introduction of a homogeneous

---

1. The other [illegible] the coin, namely how algorithm analysts can help statisticians, is examined by Shamos [1976].

model for random problems, which should help prevent the kinds of misinterpretations which have appeared in initial efforts to deal with probabilistic models; promotion of the idea that algorithms need not always get the exact answer in order to be viable; and introduction of long-ignored probabilistic and statistical tools to enable design and analysis of algorithms under very general probabilistic assumptions.

The introductory chapter begins with a brief review of previous work in the area, although most of the details are left for later chapters. Section 1.2 is a summary of Chapter 2, on stochastic convergence and probabilistic algorithms; Section 1.3 is a summary of Chapter 3, on randomization and sampling; and Section 1.4 is a summary of Chapter 4, on the uses of order statistics. Section 1.5 introduces some basic material which is essential to developing the relationships between computer science and statistics. There is a description of notation and basic probability theory and a unifying concept of "random structures" which will be used throughout the thesis. Finally, in Section 1.6 we mention some open problems and present several points which the reader should keep in mind as he reads the more technical material of later chapters.

1.1. Previous Work

Only within the last few years have serious attempts been made to investigate probabilistic models of algorithm behavior. One of the most revolutionary ideas, at least to computer scientists and mathematicians, is the notion offered by Karp [1976] and Rabin [1976], among others, that an algorithm need not get the exact answer for every input. While it is, of course, most desirable that an algorithm always get the correct solution, this is not necessarily the most cost-effective approach, since it is widely believed that solving NP-hard problems exactly requires exponential computing time.

There are three options available to help overcome this difficulty. First, it is possible for an algorithm to produce a good approximation all the time, an alternative which has been recognized for quite a while. Karp [1976] credits Graham [1966] with pioneering such algorithms for NP-hard problems. Even producing a guaranteed good approximation, however, is NP-hard for certain problems (see Garey and Johnson [1976]). A second possibility is for an algorithm to get the exact answer most of the time (Rabin [1976]). Finally, an algorithm could produce a good approximation to the correct answer most of the time (Karp [1976]). A short survey article by Gimady, Glebov, and Perepelica [1976] cites similar ideas in the Russian mathematical literature.

Unfortunately, there is now considerable confusion regarding definitions of certain key terms being used to describe probabilistic algorithms, such as Karp's [1976] algorithm for the Euclidean traveling salesman problem and Posa's [1976] algorithm for the Hamiltonian circuit problem. That this confusion exists is undeniable, even though it is not mentioned in the literature. The source of the difficulty seems to be that there are at least three non-equivalent probabilistic models (and associated definitions of the phrase "almost everywhere"), but results obtained under one model are commonly cited in papers using a different one. Chapter 2 deals with this problem, showing the relationships among the different models and how some proofs could be revised to take advantage of these relationships. Although this may sound like a serious attack on the authors or their results, it is just the opposite. The fact that some subtle problems have been overlooked in such pioneering efforts is not unusual from a historical viewpoint. An opportunity to clear them up at their inception is too good to miss.

Although there have been many expected-time analyses of discrete algorithms,

most authors make very restrictive assumptions about the distributions of input parameters. Two notable exceptions are Spira [1973], who assumes no particular distribution of edge weights for the shortest path problem, but merely that they are independent; and Bentley and Shamos [1978], who make only a very weak technical assumption about the distribution of the points to prove good expected behavior of their planar convex hull algorithm. One of the many open questions in this area is to develop a problem model which allows dependence among probabilistic quantities, and then analyze it, which seems feasible since mild dependence is allowed by a variety of statistical theorems.

Hoare [1962] proposed that analysis of Quicksort could be freed of the assumption of equally likely input permutations simply by choosing the partitioning element at random. More recently Yuval [1975b], Rabin [1976], and Carter and Wegman [1977] have suggested that this idea be used to design algorithms which perform well under a wide range of probabilistic assumptions. In Chapter 3 we continue this trend, and show how randomization and sampling can affect both the design and analysis of many algorithms.

Chapter 4 includes new probabilistic models for some problems which can be analyzed by the use of order statistics. Borovkov [1962], Weide [1976], Golden [1977], Baudet [1978], and Robinson [1978] have previously made use of such methods for computer science problems. Some of the general theorems in Chapter 4 have as corollaries a number of special results regarding the asymptotic behavior of the solutions to optimization problems on graphs. The previous cases have been analyzed on an ad hoc basis, depending on the specific problem and on the distribution of edge weights.

1.2. Summary of Chapter 2: Stochastic Convergence and Probabilistic Algorithms

Chapter 2 is a discussion of the analysis of "probabilistic approximation algorithms" using the concepts of stochastic convergence defined in Section 1.5.2. Such an algorithm usually, but not necessarily always, produces the exact solution to a problem or at least a good approximate answer. We would like to be able to characterize a particular probabilistic approximation algorithm by a statement of the form, "The algorithm produces an answer having relative error at most ϵ with probability at least p." A good probabilistic approximation algorithm would have a small value of ϵ and p near one.

Unfortunately, most problems for which probabilistic approximation seems appropriate (such as the NP-hard problems) cannot be solved even in this weak sense by simple algorithms. As a result, the probabilistic analysis of such an algorithm is typically complicated by the fact that certain steps of the algorithm are not independent, and by the fact that the answer produced by the algorithm is not independent of the true answer. The latter correlation is, of course, desirable (otherwise we would be hard pressed to justify the procedure as an algorithm for solving the problem), but it contributes to making probabilistic analysis extremely difficult in general.

As an alternative, we follow Karp [1976] in proposing that the behavior of a probabilistic approximation algorithm be characterized by the stochastic convergence (to zero) of the sequence of errors in the answers it produces to a sequence of random problems. We show how to deal with dependence among the relevant random variables, and introduce the notions of "strong" and "weak" success of algorithms to

describe those which have error sequences which converge to zero almost surely and in probability, respectively.

It is obvious that the probabilistic model of the problem instances can be an important factor in determining how "strongly" an algorithm succeeds in this sense. If edge weights in a graph are chosen from a uniform distribution, for example, the algorithm might succeed strongly, whereas if they are chosen from a normal distribution the algorithm might not work at all. This possibility is apparently recognized by everyone. It turns out that a much more subtle problem with probabilistic models has gone unnoticed in the literature of the field, and we propose a scheme to correct this deficiency.

The solution involves the distinction between what we call the "incremental problem model", in which the n^{th} problem of the sequence differs only incrementally from the previous one, and the "independent problem model", in which the n^{th} problem of the sequence is totally independent of the previous one. Theorem 2.8 is our main result relating the problem models and modes of stochastic success, and demonstrates that strong success in the independent model is a strictly stronger criterion than strong success in the incremental model, which is strictly stronger than weak success in either model.

Unfortunately, of the many probabilistic analyses of algorithms which demonstrate strong success, most apply only in the incremental model. Section 2.3 gives a review of many of the papers in this area in an attempt to illustrate the confusion which can arise if the distinction between problem models is not made explicitly. We therefore propose that this difference be recognized, and argue for adoption of the independent problem model as the canonical basis for proving stochastic success of probabilistic approximation algorithms.

Finally, in Section 2.5 we give a detailed analysis of Karp's [1976] algorithm for the Euclidean traveling salesman problem. Theorem 2.9 suggests that the algorithm succeeds strongly in the independent model, although that conclusion has yet to be proved. The long and rather detailed proof of Theorem 2.9 is included to demonstrate the use of several techniques which seem to be of universal utility in dealing with such problems.

Chapter 2 is by far the most difficult reading in this thesis, and its importance to subsequent results rests primarily with the definitions of strong and weak success and the identification of the two problem models. The reader who is familiar with these concepts and understands the hierarchy described in Theorem 2.8 should have no difficulty interpreting later results which refer to Chapter 2.

1.3. Summary of Chapter 3: Randomization and Sampling

Chapter 3 contains many practical techniques and results. We begin with a classification of algorithms into "probabilistic approximation algorithms" of Chapter 2, "randomized algorithms", and all others. A randomized algorithm is non-deterministic in the sense that it may not perform exactly the same computation if given the same inputs. The non-determinism is the result of a randomization or sampling step in the algorithm which is designed to give the algorithm good expected behavior over a wide range of input distributions. A natural correspondence between these algorithm classes and parametric and non-parametric statistics is introduced in order to suggest which statistical techniques may be most useful in designing certain types of algorithms.

Section 3.3 is part of the justification for the title of the thesis. We introduce

four general techniques for designing randomized and probabilistic approximation algorithms using ideas from statistics. These include "randomization", which includes as a special case the process of shuffling the input prior to running an algorithm in an attempt to achieve good expected running time regardless of the input permutation; "approximation in rank", which is useful in problems on totally ordered sets and for discrete optimization problems; "estimation and subsampling" for designing probabilistic approximation algorithms; and the use of "empirical distribution functions" to extend the domain of good behavior of certain algorithms from uniformly distributed inputs to all distributions satisfying certain technical conditions.

In Section 3.4 we present at least two examples of the use of each technique. An algorithm for sorting real numbers from a wide class of distributions in linear expected time is given in Section 3.4.1. Empirical results show that the algorithm, Binsort, is a practical alternative to Quicksort when more than a few hundred items are to be sorted. We also include new on-line algorithms for selection problems which use very little space, but therefore are necessarily only approximate. Again, empirical results indicate that the approximations are better in practice than can be proved in theory.

The techniques are also shown to be effective in designing and analyzing algorithms for discrete optimization problems and for geometrical problems. They lead to new algorithms for some closest point problems which run in linear expected time for a large class of point distributions. Other geometrical problems, such as finding the convex hull of a set of points in the plane, can be solved in linear expected time by using Binsort to do sorting. The expected running time of any algorithm for which the worst case is dominated by a sorting step can often be improved by this method.

1.4. Summary of Chapter 4: Order Statistics

Some intriguing results from the field of order statistics are used in Chapter 4 to analyze the behavior of solutions to graph optimization problems and to compare these with the behavior of greedy algorithms for such problems. The results of Section 4.2.1 include a host of previous results in the literature as special cases. In particular, Theorems 4.8 and 4.9 relate the value of the optimal solution to a problem defined by an objective function on the edge weights to the existence in a random graph of a subgraph satisfying the structural constraints of the problem. For instance, they relate the length of the optimal traveling salesman tour in a randomly weighted complete graph to the existence of a Hamiltonian circuit in a random graph.

There is also a discussion of the rather surprising fact that a randomized algorithm (as defined in Chapter 3) can, in theory at least, possibly be improved simply by starting up several instantiations of the same problem simultaneously and "time-sharing" the computing resources among the different versions. We state a condition on the distribution of running times of a randomized algorithm which, if satisfied, assures that the algorithm does not have optimal expected running time.

Finally, a few easy results from order statistics are used in the analysis of a schema for problem decomposition for asynchronous multiprocessors. The results are extended from the case of an ideal multiprocessor to one in which there is overhead associated with the scheduling and dispatching of tasks to the processors.

1.5. Background Material

This section contains a summary of notation, definitions, and elementary probability theory which will be used throughout the remaining chapters. It is intended

only to provide a basis for the models and terminology which we will propose and use later. Most of the new concepts are defined when they arise naturally in later chapters, so only common terms and ideas are reviewed here. While much of the discussion may seem unnecessarily formal, subsequent issues will be much easier to identify with this foundation.

1.5.1. Notation

When dealing with asymptotic behavior of functions, our notation will essentially follow that used by Knuth [1976]. Specific notations are available to describe the relationships between functions f(n) and g(n), all of which are based on the behavior of the ratio f(n)/g(n) for all sufficiently large values of n. We say that

$f(n) = o(g(n))$ iff $f(n)/g(n) \to 0$.

$f(n) = O(g(n))$ iff $f(n)/g(n) \leq c$ for some constant c.

$f(n) = \Omega(g(n))$ iff $f(n)/g(n) \geq c$ for some constant $c > 0$.

$f(n) = \theta(g(n))$ iff $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.

Another possibility, $f(n)/g(n) \to \infty$, is not specifically accounted for by this notation, but it turns out that it would be especially useful to have some way of indicating this behavior. By symmetry, the correct choice would seem to be $f(n) = \omega(g(n))$. Rather than adopt this non-standard notation we will simply say that f(n) grows faster than g(n) if this condition is satisfied.

Other notation is essentially standard. For example, $x \to A^+$ means that x approaches A from above, and $F(x^-)$ is the limit of F(z) as z approaches x from below.

Most of the other terminology commonly used in analysis of discrete algorithms is also used here; for example, log means logarithm to the base 2. See Weide [1977] for similar conventions and descriptions of most of the problems which will be examined here.

1.5.2. Basic Probability Theory

Perhaps the major problem which plagues attempts to use probability and statistics in diverse applications areas is the limited degree to which these ideas are typically developed. One of the goals of this thesis is to define clearly the problem models being analyzed and to attempt to put previous work on a sound footing.

The basis of the probability theory we will need is the probability space (Ω,B,P). The set Ω is called the sample space, and consists of elements $\omega \in \Omega$ called sample points. B is a σ-field, or σ-algebra, or Borel field, of Ω, which means that it is a class of subsets of Ω which is closed under complementation and countable union (and as a result of these two, also under countable intersection). Finally, P is a probability measure; that is, P satisfies the following axioms:

(1) $P(\Omega) = 1$.

(2) $P(E) \geq 0$ for every $E \in B$.

(3) $P(\bigcup_n E_n) = \sum_n P(E_n)$ whenever $E_i \cap E_j = \phi$ (the null set) for every $i \neq j$.

Given a probability space (Ω_0,B_0,P_0), we define the infinite product space (Ω,B,P) in which $\Omega = \Omega_0 \times \Omega_0 \times \ldots$, and where B is the usual σ-field and P the usual

product measure there (see Halmos [1950] for more details). A sample point $\vec{\omega} \in \Omega$ is an infinite sequence $(\omega_1, \omega_2, \ldots)$ where $\omega_n \in \Omega_0$. Such a space turns out to be very useful in several respects, and every space (Ω,B,P) will hereafter be assumed to be this infinite product space. This is very important, since some lemmas and theorems will not make sense in a general probability space.

Random variables are measurable real-valued functions on the sample space Ω. A random variable X has a distribution function $F(x) = P\{ \vec{\omega}: X(\vec{\omega}) \leq x \}$.[2] A distribution function is right-continuous (i.e., $F(x^+) = F(x)$) for all values of x, but may not be left continuous at a countable number of points where it has jump discontinuities. Since F(x) is non-decreasing, it has an inverse $F^{-1}(y) = \inf\{ x: F(x) \geq y \}$ (see Chung [1974]).

The events $E_1, E_2, \ldots, E_n$ are totally independent iff $P\{\bigcap_{1\leq i\leq n} E_i\} = \prod_{1\leq i\leq n} P\{E_i\}$. The sequence of random variables $\{X_n\}$ will be said to be independent if no two of the functions $\{X_n\}$ depend on common components of a sample point $\vec{\omega}$. This is a non-standard definition of independence of random variables, but clearly, whenever $\{E_n\}$ are events involving, respectively, the independent random variables $\{X_n\}$, the events $\{E_n\}$ are totally independent. The random variables $\{X_n\}$ are identically distributed if the distribution of X_n does not depend on n. If $\{X_n\}$ have the same distribution as another random variable X, we write $X_n \sim X$. Random variables which satisfy both these conditions are independent and identically distributed (i.i.d.).

---

2 . Even though P is a function, it is customary to omit the parentheses around its argument, which is an event or set and is usually delimited by braces.

The expected value of a function of a random variable X, say g(X), is denoted E(g(X)). It is defined to be $\int g(x)dF(x)$ whenever $\int |g(x)|dF(x)$ exists, where F is the distribution function of X. The mean, or expected value of X, is simply E(X). The variance of X, denoted D(X), is just $E((X - E(X))^2) = E(X^2) - E(X)^2$.

The normal distribution $F(x) = (2\pi\sigma^2)^{-1/2}\int_{-\infty}^{x} \exp(-(t - \mu)^2/(2\sigma^2))dt$ is given the special name $\mathfrak{N}(\mu,\sigma^2)$. A random variable X having this distribution has mean μ and variance σ^2. By a slight abuse of notation, this fact will be denoted $X \sim \mathfrak{N}(\mu,\sigma^2)$.

Of primary importance in later chapters will be stochastic convergence of a sequence of random variables. The sequence of random variables $\{X_n\}$ converges almost surely, or almost everywhere, or with probability one, to X (written $X_n \rightarrow_{as} X$) whenever $P\{ \vec{\omega}: \lim X_n(\vec{\omega}) = X(\vec{\omega}) \} = 1$. Similarly, the sequence $\{X_n\}$ converges in probability, or in measure, to X (written $X_n \rightarrow_{pr} X$) whenever, for all $\epsilon > 0$, $\lim P\{ \vec{\omega}: |X_n(\vec{\omega}) - X(\vec{\omega})| < \epsilon \} = 1$. Finally, $\{X_n\}$ converges in distribution to X (written $X_n \rightarrow_{d} X$) whenever $F_n(x) = P\{ \vec{\omega}: X_n(\vec{\omega}) \leq x \}$ converges to $F(x) = P\{ \vec{\omega}: X(\vec{\omega}) \leq x \}$ at all continuity points of F.[3]

To illustrate these concepts with an example of an infinite product space, we

---

3. Convergence in distribution is actually a property of the sequence $\{F_n\}$, so that the random variables $\{X_n\}$ need not be defined on the same space. This technical point is of no real concern to us, since all random variables used here will be defined on the same space $\mathfrak{A}$ described below.

cite perhaps the most useful instance of all. The sample space Ω_0 is the set of reals $0 \le \omega_n < 1$, F_0 is the usual Borel field on [0,1), and P_0 is the usual Lebesgue measure. In the infinite product space, $\vec{\omega}$ is a sequence of real numbers, each of which is between 0 and 1. Furthermore, $P\{\omega_n \le x\} = x$ for $0 \le x < 1$. In more common terms, each ω_n is uniformly distributed between 0 and 1, and a sequence $\vec{\omega}$ consists of independently uniformly distributed components. This space is so special that we will call it by its own name, $\mathfrak{A}$.

By means of the so-called probability-integral transformation, it is possible to define a random variable X having any given distribution function F (not necessarily continuous) by using the probability space $\mathfrak{A}$, and the functional inverse of F. Briefly, because F is increasing, $F(x) = P\{ X \le x \} = P\{ F(X) \le F(x) \}$. Letting $X = F^{-1}(\omega_i) = \inf\{ x: F(x) \ge \omega_i \}$ for some i gives $P\{ \omega_i \le F(x) \} = F(x)$, which is satisfied exactly when ω_i is uniformly distributed between 0 and 1, as it is in the space $\mathfrak{A}$. This principle is used by simulation systems to generate random numbers from an exponential distribution, for example, by computing a function of random numbers from the uniform distribution. Thus, the random variables $X_n(\vec{\omega}) = -\ln(1 - \omega_n)$ are i.i.d., each having an exponential distribution.

The primary advantage of defining probability-theoretic terms by using the infinite product space is the elegant manner in which these terms appear. There is no need to talk about balls in urns, or other combinatorial or procedural structures, in order to define or understand such diverse topics as independence and stochastic

convergence. Even better, such special models can be defined within the probability-space model in a natural way, a fact which will enable us to see clearly the sources of difficulty in a number of misinterpretations which have recently appeared in the literature.

1.5.3. Random Structures

Since random variables are functions on Ω, they may be defined in arbitrarily complex ways, including functional composition. Specifically, $X(\vec{\omega})$ may be based on an intermediate random structure. The structure is determined by the argument $\vec{\omega}$, and then the final value of X is determined by the structure. In this section, we will examine some random structures based on the space $\mathfrak{A}$ which arise in computer science problems.

The simplest such structure is a natural extension of the idea of a random variable. An ordered list of random variables is a random vector, or random point.[4] Suppose that we were interested in the distances from the origin of points uniformly distributed in the unit cube. Our final random variable X might be defined as $X(\vec{\omega}) = (\omega_1^2 + \omega_2^2 + \omega_3^2)^{1/2}$. For this problem, the random vector is "hidden" by the fact that it is supposed to be uniformly distributed in the cube. More generally, if we were interested in points from a given multivariate distribution, then we would use $X(\vec{\omega}) = (x_1^2 + x_2^2 + x_3^2)^{1/2}$, where (x_1 , x_2 , x_3) is a random vector determined by transforming

---

4. This should not be confused with a sample point $\vec{\omega}$.

some components of $\vec{\omega}$ to produce the desired distribution of points. Random vectors are useful in probabilistic models of problems from geometry, mathematical programming, polynomial arithmetic, etc.

Another structure which frequently appears in computer science problems is the random permutation. We typically would like a model under which each of the n! possible permutations of n objects is equally likely. A random variable of possible interest is the number of comparisons required to sort n elements of a linearly ordered set, whose expected value we wish to compute under this probability model. It is usually easy to think in procedural terms when generating random structures (see Sedgewick [1977] for more about permutations). In this case, the permutation π defined by $\vec{\omega}$ can simply be that ordering of the integers $\{1,2,\ldots,n\}$ for which $\omega_{\pi_1} < \omega_{\pi_2} < \ldots < \omega_{\pi_n}$.

The most complex of the random structures which we will encounter are random graphs (see Erdos and Renyi [1959] and Erdos and Spencer [1974]). A classical problem from random graph theory is the question of connectivity: What is the probability that a random labelled graph with n vertices and m edges is connected? We can define a random variable X of the 0-1 type (0 if the graph is not connected, 1 if it is connected), and find its expected value, assuming that every labelled graph with n vertices and m edges is equally likely.[5] Again, a procedural description of X is easy. Consider the first $\binom{n}{2}$ components of $\vec{\omega}$ to be numbered:

$$\begin{array}{ccccc} \omega_{1,2} & \omega_{1,3} & \omega_{1,4} & \cdots & \omega_{1,n} \\ & \omega_{2,3} & \omega_{2,4} & \cdots & \omega_{2,n} \end{array}$$

---

5. Other definitions of random graphs are possible. See Chapter 4.

and so forth, through $\omega_{n-1,n}$. If y is the m^{th}-smallest of these components, then we simply let edge (i,j) be present in the graph iff $\omega_{i,j} \leq y$. Now X is 0 if the resulting graph is not connected and 1 if it is.

Slight extensions of ordinary random graphs are random directed graphs and random weighted graphs. The latter are especially useful in modeling certain mathematical programming problems, since the weights may be assigned to vertices, or edges, or both, and may have arbitrary distributions.

Up to this point, we have not made use of the fact that $\vec{\omega}$ is infinite-dimensional, since each random structure depends on only a finite number of components of $\vec{\omega}$. However, it is easy to use $\vec{\omega}$ to define sequences of random structures, and thereby sequences of random variables for which we can explore the properties of stochastic convergence. There are at least two different ways of defining such sequences, one of which "re-uses" the initial components of $\vec{\omega}$ to determine each structure, and the other of which "discards" the used components.[6] The structures, and hence the random variables, described by the former method are not independent, whereas those defined by the latter method are independent. Random variables based on these two different sequences of random structures can exhibit different modes of stochastic convergence.

---

6 . This difference, as far as I can determine, has been overlooked by virtually everyone using sequences of random structures, but is very important. Chapter 2 explains the ramifications of the distinction.

1.6. Conclusions and Further Work

Several statistical techniques are shown to be useful in the design and analysis of algorithms for computer science problems. The techniques illustrated here, however, are only the simplest used by statisticians. In the case of sampling, for instance, much more sophisticated schemes than the random sampling used in our algorithms can be devised. It remains to be seen which advanced statistical tools can be profitably applied to algorithm design and analysis.

In addition to this general observation, there are several other questions of varying degrees of importance which could be explored in an extension of this work or which remain as open problems. The following is a partial list which the reader may keep in mind as he continues through Chapters 2, 3, and 4.

(1) There are several problems for which it is known that finding an approximate solution with bounded relative error is NP-complete. Is there any problem for which it can be demonstrated that finding an approximate solution with bounded relative error almost surely is NP-complete, in some reasonable probabilistic model?

(2) Prove the missing companion to Theorem 2.9 and show that the result of Beardwood, et al. [1959] holds almost surely in the independent model. Such a proof would, according to Theorem 2.9(b), assure that Karp's [1976] original algorithm for the TSP succeeds strongly in the independent problem model.

(3) Extend the techniques of Section 3.3.4 from distributions of bounded random variables to a more general class of distributions. This looks somewhat easier than it probably is because the present form of the algorithm "solve" relies on the fact that the expected number of items in any bin is bounded by a

constant, and this might not be true if the minimum and maximum elements could wander off abritrarily far.

(4) Develop a technique for proving lower bounds in a probabilistic setting. (The work of Yao [1977] seems important here.) As a starting point, prove that an on-line selection algorithm displaying the behavior described in Theorem 3.14 must use at least as much space as the procedure "median_est".

(5) Show how to prove non-trivial lower bounds in a model of computation which includes the floor function.

(6) Perform some experiments to evaluate the technique proposed in Section 3.4.3 for λ-opt heuristics.

(7) Extend the results for geometry problems in the plane to higher dimensions. Some of them generalize very naturally, but others do not. In particular, the algorithm for constructing the Voronoi diagram does not seem to extend naturally to three or more dimensions and continue to run in linear expected time.

(8) Using Lemma 4.4 as a starting point, derive the three possible forms of limit distributions for extreme values of random variables. The classical approach to this problem (see David [1970] for a description) uses a very elegant argument to show what limiting distributions are possible, but does not make use of the result of Lemma 4.4. Because of its simplicity, this lemma seems like an appropriate candidate for the seed of an alternative proof.

(9) Find a companion to Theorem 4.8 which gives an almost sure lower bound on the value of the optimal solution.

(10) Exhibit an uncontrived algorithm for a real problem which can be improved by

using the method of Section 4.2.2. In the event that one is found, suggest an extension to an existing programming language which permits the programmer to have control over the scheduling of parallel tasks.

(11) Test the conclusions of Section 4.2.3 for a real problem on a real multiprocessor. Preliminary measurements of an integer programming code running on C.mmp inspired the investigation of this problem in the first place, and tend to confirm the conclusions. However, each problem required such an enormous amount of computer time that no statistically significant results were ever obtained.

(12) Find the variance of the solution time for the randomized algorithms presented here or, even better, the distribution of solution times. The results of Sections 4.2.2 and 4.2.3 argue that knowing the distribution might be useful in ways which are not immediately evident.

(13) Give more examples of the uses of any of the statistical techniques suggested in this thesis.

(14) Suggest other algorithm design and analysis techniques based on statistical concepts.

2. Stochastic Convergence and Probabilistic Algorithms

Three types of stochastic convergence of random variables were defined in Chapter 1. Two of these, amost sure convergence and convergence in probability, are important in describing the (stochastic) success of probabilistic approximation algorithms. By a probabilistic approximation algorithm we mean an algorithm which usually, but not always, produces the exact solution or a good approximate answer to a problem. We might characterize such an algorithm, and our knowledge of it, by dividing the class into three categories: algorithms which usually get the exact answer, those which usually get within some error criterion ϵ, and those which have a certain error distribution F_ϵ. Most of the algorithms we will consider are of the second type, although it is desirable but only occasionally possible to find the error distribution.

The convergence concepts themselves are first examined in some detail, after which we discuss their applications to specific probabilistic approximation algorithms such as those described by Karp [1976].

2.1. Stochastic Convergence

Although convergence in distribution is used in some proofs we will need, the other two modes of stochastic convergence are of more immediate interest. Recall that

a sequence of random variables $\{X_n\}$ converges almost surely[1] to the random variable X ($X_n \rightarrow_{as} X$) iff $P\{ \vec{\omega}: \lim X_n(\vec{\omega}) = X(\vec{\omega}) \} = 1$. Similarly, the sequence $\{X_n\}$ converges in probability[2] to X iff, for every $\epsilon > 0$, $\lim P\{ \vec{\omega}: |X_n(\vec{\omega}) - X(\vec{\omega})| < \epsilon \} = 1$.

From both an intuitive and a practical standpoint, it is profitable to view stochastic convergence in terms of individual sample points $\vec{\omega}$. The sequence of random variables $\{X_n\}$ converges almost surely to X if the set of sample points for which the sequence of real numbers $\{X_n(\vec{\omega})\}$ converges to $X(\vec{\omega})$ has probability one. It converges in probability if, for every $\epsilon > 0$, $X_n(\vec{\omega})$ is within ϵ of $X(\vec{\omega})$ for sets of sample points whose probabilities approach one as $n \rightarrow \infty$.

In the latter case, it is possible that the sequence of reals $\{X_n(\vec{\omega})\}$ does not converge to $X(\vec{\omega})$ for any sample point $\vec{\omega}$. Consider a case where X_n are 0-1 random variables, and X is identically 0. If $X_n \rightarrow_{as} X$ then for each sample point $\vec{\omega}$ in a set of probability one, the sequence $\{X_n(\vec{\omega})\}$ can have only finitely many "ones". For $X_n \rightarrow_{pr} X$, however, "ones" may continue to appear occasionally (i.e., infinitely often) in every sequence $\{X_n(\vec{\omega})\}$, although for most such sequences they cannot appear too frequently.

An example of a sequence which converges in probability but not almost surely

---

1 . The analogous concept in real function theory uses the phrase "almost everywhere". Since we are dealing with probability, though, the terms "almost surely" and "with probability one" are preferable.

2 . Again, this terminology is preferred to the phrase "in measure" for our purposes.

is provided by $X_n(\vec{\omega}) = 1$ if $\omega_n < 1/n$ and $X_n = 0$ otherwise. It is clear that "ones" can appear infinitely often in this sequence, but it is not obvious that this actually happens for a set of sample points of positive measure. However, Lemma 2.2 shows that convergence of this sequence is not almost sure, so that the probability that the sequence does not converge is strictly positive. On the other hand, the probability of a "one" appearing in position n tends to 0 as $n \to \infty$, so the sequence does converge in probability.

It is clear from these intuitive pictures that almost sure convergence implies convergence in probability, and this can be proved rigorously.

LEMMA 2.1 -

If $X_n \to_{as} X$ then $X_n \to_{pr} X$.

Proof - See Chung [1974], page 66. □

It is easier in practice to prove convergence in probability, directly from its definition, than it is to prove almost sure convergence. Fortunately, there is an alternate characterization of almost sure convergence, provided by

LEMMA 2.2 - (Borel-Cantelli Lemma) -

If $\sum_n P\{ |X_n - X| > \epsilon \}$ is finite for every $\epsilon > 0$, then $X_n \to_{as} X$. If $X_n \to_{as} X$ and the $\{X_n\}$ are independent, then $\sum_n P\{ |X_n - X| > \epsilon \}$ is finite for every $\epsilon > 0$.

Proof - See Chung [1974], pages 73-78. □

From the definition of convergence in probability and the Borel-Cantelli Lemma, we can prove

LEMMA 2.3 -

Let $\{X_n\}$ be independent.

(a) $X_n \rightarrow_{pr} X$ iff, for every $\epsilon > 0$, $P\{ |X_n - X| > \epsilon \} = o(1)$.

(b) Let $\log^{(i)}n$ be the i^{th} iterated logarithm of n (that is, let $\log^{(0)}n = n$ and $\log^{(i)}n = \log \log^{(i-1)}n$); and let $f_k(n) = \left(\prod_{0 \le i < k} \log^{(i)}n \right)^{-1}$, with $f_0(n) = 1$. If $X_n \rightarrow_{as} X$, then $P\{ |X_n - X| > \epsilon \} = o(f_k(n))$ for every $\epsilon > 0$ and every $k \ge 0$. If, for every $\epsilon > 0$, $P\{ |X_n - X| > \epsilon \} = O(f_k(n) \cdot (\log^{(k)}n)^{-(1+\delta)})$ for any $k \ge 0$ and any $\delta > 0$, then $X_n \rightarrow_{as} X$.

Proof - Part (a) is just a restatement of the definition of convergence in probability. Part (b) follows from the Borel-Cantelli Lemma, as follows.

Note first that $\sum_n f_k(n)$ diverges for all $k \ge 0$, since $\sum_{i \le n} f_k(i) = \theta(\log^{(k)}n)$, which can be proved by comparing the sum to the integral $\int f_k(x)dx$. Thus, if $\sum_n P\{ |X_n - X| > \epsilon \}$ is to be finite, $P\{ |X_n - X| > \epsilon \}$ must approach zero faster than $f_k(n)$ for every $k \ge 0$. On the other hand, $\sum_n f_k(n) \cdot (\log^{(k)}n)^{-(1+\delta)}$ is finite for every $k \ge 0$ and every $\delta > 0$, which can also be proved by considering the corresponding integral (see, for instance, Hardy [1924]). This demonstrates statement (b) of the lemma. □

Lemmas 2.1 and 2.3 clearly illustrate that $X_n \rightarrow_{as} X$ is a (possibly strictly) stronger statement than $X_n \rightarrow_{pr} X$. Even so, it is not strong enough that we can always draw even the seemingly innocent conclusion that $E(X_n) \rightarrow E(X)$.

LEMMA 2.4 -

If $\{X_n\}$ are uniformly bounded by a constant, then $X_n \rightarrow_{pr} X$ implies $E(|X_n - X|^p) \rightarrow 0$ for every $p > 0$. In general, however, even $X_n \rightarrow_{as} X$ does not imply that $E(|X_n - X|^p) \rightarrow 0$.

Proof - The first part of the lemma follows directly from Theorem 4.1.4 of Chung [1974]. An example of a case where $\{X_n\}$ are not uniformly bounded can be constructed using the space $\mathfrak{A}$. Let $X_n(\vec{\omega}) = 0$ if $\omega_n > 1/n^2$, and $X_n(\vec{\omega}) = 2^n$ if $\omega_n \leq 1/n^2$. Applying the Borel-Cantelli Lemma, we see that $X_n \rightarrow_{as} 0$, but $E(|X_n|^p) = 2^{np}/n^2 \rightarrow \infty$ for all $p > 0$. □

Another somewhat non-intuitive aspect of stochastic convergence is identified by the following lemma, which uses the difference between identical random variables and identically distributed random variables.

LEMMA 2.5 -

Let $\{Z_n\}$ be independent, and let $h(n)$ be an integer-valued function which is non-decreasing and unbounded. Define $\{X_n\}$ as a sequence of random variables with the property that $X_n = Z_{h(n)}$. Define $\{Y_n\}$ as a sequence of independent random variables with the property that $Y_n \sim Z_{h(n)}$.

(a) If $Z_n \rightarrow_{as} Z$ then $X_n \rightarrow_{as} Z$.

(b) If $Z_n \rightarrow_{as} Z$ then $Y_n \rightarrow_{pr} Z$, but it is not necessarily true that $Y_n \rightarrow_{as} Z$.

Proof - We may assume without loss of generality that $Z = 0$ by considering the random variables $Z_n - Z$, $X_n - Z$, and $Y_n - Z$. To prove part (a), note that for each $\vec{\omega}$, $\{Z_n(\vec{\omega})\}$ and $\{X_n(\vec{\omega})\}$ have a common subsequence, call it $\{C_n(\vec{\omega})\}$. The sequence $\{X_n(\vec{\omega})\}$ contains only the terms $\{C_n(\vec{\omega})\}$, possibly repeated, depending on h of course. In particular, we have $X_n(\vec{\omega}) = C_1(\vec{\omega})$ for $1 \le n \le m_1$, where $m_1 = \min\{n: h(n) > h(1)\}$; similarly, $X_n(\vec{\omega}) = C_2(\vec{\omega})$ for $m_1 < n \le m_2$, where $m_2 = \min\{n: h(n) > h(m_1)\}$; and so forth.

Now if $Z_n(\vec{\omega}) \to 0$, then $C_n(\vec{\omega}) \to 0$, since $\{C_n(\vec{\omega})\}$ is a subsequence of $\{Z_n(\vec{\omega})\}$. As we have seen above, $C_n(\vec{\omega}) \to 0$ implies that $X_n(\vec{\omega}) \to 0$. Therefore, $X_n(\vec{\omega}) \to 0$ whenever $Z_n(\vec{\omega}) \to 0$, and the latter occurs for every $\vec{\omega}$ in some set of probability one. Thus, $X_n \to_{as} 0$.

Part (b) is somewhat different, since the only relationship between $\{Y_n\}$ and $\{Z_n\}$ is that $P\{ Y_n \le x \} = P\{ Z_{h(n)} \le x \}$. Since $\{Z_n\}$ are independent, Lemma 2.3 applies, showing that $P\{ |Z_n| > \epsilon \} \to 0$ for every $\epsilon > 0$. Hence, $P\{ |Y_n| > \epsilon \} \to 0$ since h(n) is non-decreasing and unbounded, proving that $Y_n \to_{pr} 0$.

As a counterexample to the claim that $Y_n \to_{as} 0$, use $\mathfrak{A}$ to define $Z_n(\vec{\omega}) = 0$ if $\omega_n > 2^{-n}$ and $Z_n(\vec{\omega}) = 1$ if $\omega_n \le 2^{-n}$, and let $h(n) = \lfloor \log n \rfloor + 1$. For all $\epsilon > 0$ we have $P\{ |Z_n| > \epsilon \} \le 2^{-n}$, so by the Borel-Cantelli Lemma, $Z_n \to_{as} 0$. On the other hand, for $0 < \epsilon < 1$, $P\{ |Y_n| > \epsilon \} \ge 1/(2n)$. Again, by the Borel-Cantelli Lemma, $\{Y_n\}$ does not converge almost surely. □

In the computer science literature, convergence concepts have been tied to

algorithm behavior in the sense of probabilistic approximation. Typically, a sample point $\vec{\omega}$ describes a sequence of <u>random problems</u> (based on random structures such as those introduced earlier), one of each size n for $n \geq 1$. The random variable X_n is the error produced by the algorithm on a problem of size n; in particular, $X_n(\vec{\omega})$ is the error when the algorithm is applied to the problem of size n specified by $\vec{\omega}$. Under these circumstances, we will say that the algorithm <u>succeeds strongly</u> if $X_n \rightarrow_{as} 0$, and that it <u>succeeds weakly</u> if $X_n \rightarrow_{pr} 0$. The terms "strong" and "weak" are used by analogy to the strong and weak laws of large numbers.

The following two lemmas are helpful in proving stochastic success. The first is used when X_n is the absolute error, and the second when it is the relative error.

<u>LEMMA 2.6</u> - (Absolute error lemma) -

(a) If $Y_n \rightarrow_{pr} a$ and $Z_n \rightarrow_{pr} b$ for constants a and b, then

$$Y_n + Z_n \rightarrow_{pr} a + b.$$

(b) If $Y_n \rightarrow_{as} a$ and $Z_n \rightarrow_{as} b$, then $Y_n + Z_n \rightarrow_{as} a + b$.

<u>Proof</u> - This is a standard result, a stronger version of which is proved in Chung [1974], but the proof is instructive because it demonstrates a common technique for proving similar conclusions. For part (a), we must show that, for every fixed $\epsilon > 0$, $P\{ |Y_n + Z_n - a - b| > \epsilon \} \rightarrow 0$.

$$P\{ |Y_n + Z_n - a - b| > \epsilon \} \leq P\{ |Y_n - a| + |Z_n - b| > \epsilon \}$$

$$\leq P\{ |Y_n - a| > \epsilon/2 \text{ or } |Z_n - b| > \epsilon/2 \}$$

$$\leq P\{ |Y_n - a| > \epsilon/2 \} + P\{ |Z_n - b| > \epsilon/2 \}$$

Since $Y_n \rightarrow_{pr} a$ and $Z_n \rightarrow_{pr} b$, both terms on the right-hand side of the last inequality tend to zero as $n \rightarrow \infty$, which proves part (a). Part (b) is proved in exactly the same way, with the only difference being the insertion of summation signs $\sum_n$ before each of the probabilities. Each of the sums converges because $Y_n \rightarrow_{as} a$ and $Z_n \rightarrow_{as} b$. □

LEMMA 2.7 - (Relative error lemma) -

(a) If $Y_n \rightarrow_{pr} c > 0$ and $Z_n \rightarrow_{pr} c$, then $(Y_n - Z_n) / Y_n \rightarrow_{pr} 0$.

(b) If $Y_n \rightarrow_{as} c > 0$ and $Z_n \rightarrow_{as} c$, then $(Y_n - Z_n) / Y_n \rightarrow_{as} 0$.

Proof - Again, the proof of part (b) follows exactly the same pattern as that for part (a), so we will prove only convergence in probability. We must show that $P\{ |(Y_n - Z_n) / Y_n| > \epsilon \} \rightarrow 0$ for every fixed $\epsilon > 0$.

$$P\{ |(Y_n - Z_n) / Y_n| > \epsilon \} = P\{ |(Y_n - c - (Z_n - c)) / Y_n| > \epsilon \}$$

$$\leq P\{ |(Y_n - c) / Y_n| + |(Z_n - c) / Y_n| > \epsilon \}$$

$$\leq P\{ |(Y_n - c) / Y_n| > \epsilon/2 \text{ or } |(Z_n - c) / Y_n| > \epsilon/2 \}$$

$$\leq P\{ |(Y_n - c) / Y_n| > \epsilon/2 \} + P\{ |(Z_n - c) / Y_n| > \epsilon/2 \}$$

To show that of the probabilities on the right-hand side of the last inequality both tend to zero, we will concentrate on the first one. The proof for the other is similar. Let $0 < \delta < c$.

$$P\{ |(Y_n - c) / Y_n| > \epsilon/2 \} = P\{ |(Y_n - c) / Y_n| > \epsilon/2 \mid |Y_n - c| > \delta \} \cdot$$

$$P\{ |Y_n - c| > \delta \}$$

$$+ P\{ |(Y_n - c) / Y_n| > \epsilon/2 \mid |Y_n - c| \leq \delta \} \cdot$$

$$P\{ |Y_n - c| \le \delta \}$$

$$\le P\{ |Y_n - c| > \delta \} + P\{ |(Y_n - c) / Y_n| > \epsilon/2 \mid |Y_n - c| \le \delta \}$$

$$\le P\{ |Y_n - c| > \delta \} + P\{ |Y_n - c| > \epsilon(c-\delta)/2 \}$$

The last step follows because Y_n is positive and is, in fact, no smaller than $c - \delta$. Since both of these probabilities approach zero, the lemma is proved. □

As an example of an algorithm which succeeds in each of these modes using absolute error, consider the problem of finding the arithmetic average of n independent observations from $\mathfrak{N}(0, 1)$, a normal distribution with mean 0 and variance 1. Our probabilistic approximation algorithm will simply be to choose the first r(n) observations from the original n and average them. Assume that $r(n) = o(n)$.

Using the space $\mathfrak{A}$, we may think of the components of a sample point $\vec{\omega}$ as being numbered as in a triangular array, with n elements in row n. Row n then determines a sample[3] of size n through a suitable transformation of the components $\omega_{n,k}$. Let $X_n(\vec{\omega})$ be the difference between the average of all n observations determined by row n, $Y_n(\vec{\omega})$, and the average of the first r(n) observations, $Z_n(\vec{\omega})$. It follows immediately from known properties of the normal distribution that $Y_n \sim \mathfrak{N}(0, n^{-1})$ and $Z_n \sim \mathfrak{N}(0, r(n)^{-1})$.

---

3. A sample is a group of observations; an observation is a specific value of a random variable. In this case, a problem of size n is a sample consisting of n observations which are to be averaged.

A minor complication arises in determining the distribution of $X_n = Y_n - Z_n$, since Y_n and Z_n are not independent. We may not conclude, therefore, that $X_n \sim \mathfrak{N}\left(0 , n^{-1} + r(n)^{-1}\right)$, which would be the case if Y_n and Z_n were independent. This problem will come up again in many of the algorithms we would like to investigate. Fortunately in this instance, X_n can be written as a linear combination of Z_n and the average of the remaining n-r(n) observations, which are independent. This leads to the conclusion that $X_n \sim \mathfrak{N}\left(0 , (1-r(n)/n)^2(1/r(n)+1/(n-r(n))\right)$.

It is now a simple matter to verify that $P\{ |X_n| > \epsilon \} = \theta(r(n)^{-1/2}\exp(-\epsilon^2 r(n)/2))$. Thus, if r(n) is any non-decreasing function of n which grows without bound, $P\{ |X_n| > \epsilon \} \to 0$, so $X_n \to_{pr} 0$. In this case, the algorithm succeeds weakly. If r(n) grows faster than log n then, by Lemma 2.3, $X_n \to_{as} 0$ and the algorithm succeeds strongly. It is in fact the case that these growth conditions are necessary for weak and strong success, respectively. The reason that r(n) must be unbounded for weak success is clear. It must grow faster than log n for strong success because $\sum_n r(n)^{-1/2}\exp(-\epsilon^2 r(n)/2)$ is finite for all $\epsilon > 0$ iff that condition is met.

We could have proved the sufficiency of the conditions on r(n) more easily using Lemma 2.6, the absolute error lemma. Knowing the distributions of Y_n and Z_n, we can show that the conditions given above are sufficient for $Z_n \to_{pr} 0$ and $Z_n \to_{as} 0$, respectively. Since $Y_n \to_{as} 0$ in any event, we may conclude that $X_n \to_{pr} 0$ if r(n) grows without bound, and that $X_n \to_{as} 0$ if r(n) grows faster than log n. What we gain by not

having to worry about the independence of Y_n and Z_n is offset by the fact that we cannot say that the growth condition on r(n) is necessary. Furthermore, we know nothing about the distribution of X_n if we take the easy way out. Nevertheless, it is good to have a technique available for proving stochastic success which can easily deal with dependent random variables.

This example illustrates two prime objections to the characterization of probabilistic approximation algorithms by the "strength" of their success. In the first place, even strong success is only of theoretical value. We could let $r(n) = 1$ for $n \leq C$ and $r(n) = \log^2 n$ for $n > C$, and the above algorithm would succeed strongly for any C, even if C were larger than the size of any problem we might encounter in practice. For this reason, the approach does not enable us to conclude that an algorithm is provably practical.

Secondly, neither version of success provides any means of determining, even asymptotically, how well an algorithm works as a function of n. The only guidelines available are those provided by Lemma 2.3, which may be entirely too pessimistic. We therefore have no basis for comparison of two strongly successful algorithms, or even for one strongly successful and one weakly successful algorithm, unless the weakly successful one can be shown not to succeed strongly. Even then the weakly successful algorithm might be preferred in practice, since both characterizations are only asymptotic.

Notice that these are fundamentally different objections to the probabilistic approach than the usual one: that some assumptions are unrealistic. The assumptions

are not only reasonable, but are in fact satisfied, yet a strongly successful algorithm can remain impractical. Similarly, the objections are stronger than the typical argument against asymptotic analysis in general, which is that the conclusions are not valid except for very large values of n. In contrast to the "big-O" notation, for example, there is no way to make any meaningful quantitative statement even about asymptotic behavior. These notions of probabilistic success are therefore too weak to allow us to draw any serious conclusions regarding the practical value of an algorithm, although they do permit comparisons among algorithms, as we will see in Section 2.2.

Nevertheless, due in part to the impetus which such analysis has already received from people in the algorithms area, these basically theoretical descriptions of probabilistic approximation algorithms are here to stay. It is important that a firm and comprehensible basis for their study be established while there are not too many different results which depend on one another in a complex fashion. The goal of the next section is to point out certain difficulties which can arise, and to introduce a random problem model which can serve as the foundation for further work. In Chapter 3 we will return to the objections to stochastic convergence concepts as descriptions of success, and present some alternatives which partially overcome these arguments.

2.2. Random Problem Models

There are at least two possible reasons for wanting to deal with random problems rather than with fixed ones. The first, examined in some detail by Vajda [1972] for mathematical programming problems, is that the problem data may be uncertain; that is, the data may be random perturbations of the true values of the parameters (due to measurement errors, for instance). Among the interesting questions

in this scenario are such things as how to optimize the expected value of a linear program knowing only the joint distribution of the objective function coefficients, and not their actual values.

On the other hand, it may be that we would like to solve many instances of some type of problem, and have no prior knowledge of what these instances might be other than distributions of certain characteristics. Suppose that XYZ Company has thousands of customers throughout the United States from which it receives requests for service at random times, but has only one serviceman who is sent out every time 100 requests have accumulated. If XYZ wants to solve the resulting 100-city traveling salesman problems, company management might be interested in knowing how much extra travel cost would be incurred, on the average, by using some approximation algorithm rather than a (very costly!) exact algorithm.

Whatever the possible practical applications, random problems are considered by algorithm analysts in order to make probabilistic statements similar to those made by statisticians about statistical procedures. These statements include not only the expected resource consumption of an algorithm and the classical questions of computer science, but more recently have centered on the probability that an algorithm will achieve a specified small error. It is, of course, valuable to keep in mind the possible real-world applications of results, but in large part the models considered for analysis must be quite simple in order to be mathematically tractable. In this section, we examine problem models with respect to their suitability for discussion of stochastic success of algorithms.

A <u>random problem</u> is defined by (1) a random structure, such as set of random points or a random graph; and (2) some function of the structure, such as the mean

distance between the points or the chromatic number of the graph. In some cases, a "solution" to a problem may require not only the value of this function, but other information as well. For example, the solution to a traveling salesman problem demands an actual tour of the points and not just the length of the tour. The function value provides the basis for comparison between the exact solution and an approximation.

For dealing with stochastic success, we require sequences of random problems. A sequence can be generated in many ways from a sample point $\vec{\omega}$, but we will study only two.

The first possibility, which we call the incremental problem model, operates as follows. If $R_n(\vec{\omega})$ is the n^{th} problem determined by $\vec{\omega}$, and R_n is fully specified by k(n) components of a sample point, then $R_n(\vec{\omega})$ depends on the components $\omega_1, \omega_2, \ldots, \omega_{k(n)}$. Specifically, $R_n(\vec{\omega})$ is generated "incrementally" from $R_{n-1}(\vec{\omega})$ by some process depending on $\omega_{k(n-1)+1}, \ldots, \omega_{k(n)}$. If the underlying problem structure is a random vector, $R_n(\vec{\omega})$ might be an n-vector generated by appending one new coordinate to the (n-1)-vector $R_{n-1}(\vec{\omega})$. If it is a random graph, the incremental change might be the addition of a new vertex and some edges incident to it, with all the edges of the previous graph unchanged.

A second possibility is the independent problem model. Consider the sequence $\vec{\omega}$ to be numbered as in a triangular array, with k(n) elements in row n, where k(n) is defined as above. In this case, $R_n(\vec{\omega})$ depends on the components $\omega_{n,1}, \omega_{n,2}, \ldots, \omega_{n,k(n)}$. This means that R_n is totally independent of R_{n-1}, begin generated by all new defining

components. If the underlying problem structure is a random vector, $R_n(\vec{\omega})$ is defined by creating all new coordinate values, none of which is necessarily the same as its counterpart in $R_{n-1}(\vec{\omega})$. Similarly, if it is a random graph, an entirely new graph is created.[4]

With these two problem models, and two possible modes of stochastic success of an algorithm, there are four cases to consider. The following theorem describes the relationships between them.

THEOREM 2.8 -

(a) If an algorithm succeeds strongly in either model, then it succeeds weakly in that model, but not necessarily vice versa.

(b) If an algorithm succeeds weakly in one model, then it succeeds weakly in the other model.

(c) If an algorithm succeeds strongly in the independent problem model, then it succeeds strongly in the incremental model, but not necessarily vice versa.

Proof - Figure 2.1 shows schematically how the four possibilities are related.

Throughout the proof, let $X_n(\vec{\omega}) = f_n(\omega_1, \omega_2, \ldots, \omega_{k(n)})$ be the error of the algorithm in the incremental problem model, and let $Y_n(\vec{\omega}) = f_n(\omega_{n,1}, \omega_{n,2}, \ldots, \omega_{n,k(n)})$ be the error in the independent problem model. The difference between X_n and Y_n is that

---

4 . There are obviously "hybrid" possibilities, where there is some dependence between successive problems, but not strictly incremental generation. Such models have apparently not been used in the literature and appear to be of no value in this context.

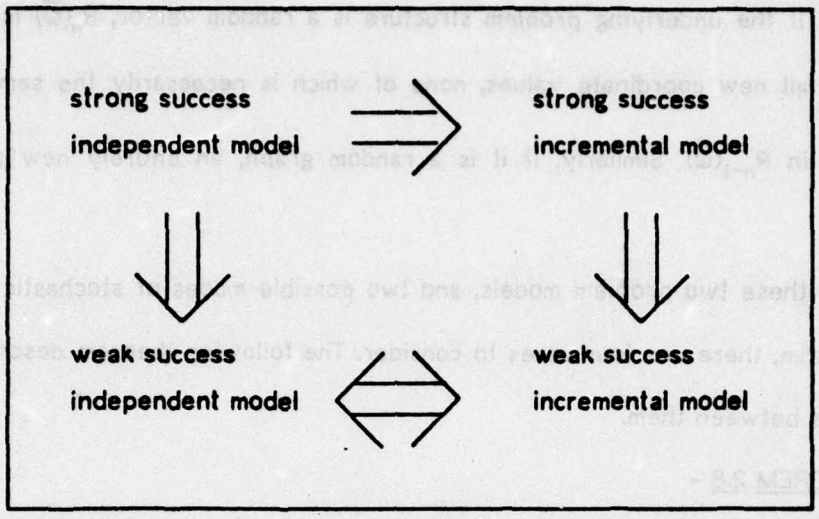


FIGURE 2.1 - Problem models and stochastic convergence.

they are the same function of different components of $\vec{\omega}$, so that the random variables $\{X_n\}$ are not independent, whereas $\{Y_n\}$ are independent.

Part (a) of the theorem is clear, since $X_n \rightarrow_{as} 0$ implies that $X_n \rightarrow_{pr} 0$, and similarly for $\{Y_n\}$. It is not the case that $X_n \rightarrow_{pr} 0$ implies that $X_n \rightarrow_{as} 0$, which is illustrated by the following example. Suppose that $k(n) = n$ and that $X_n(\vec{\omega}) = 0$ if $\omega_{k(n)} > 1/n$, and $X_n(\vec{\omega}) = 1$ if $\omega_{k(n)} \leq 1/n$. Although $X_n \rightarrow_{pr} 0$, the Borel-Cantelli Lemma shows that the convergence is not almost sure. The same example suffices for $\{Y_n\}$, proving part (a).[5]

The proof of part (b) is a direct consequence of the fact that $X_n \sim Y_n$, which is

---

5. The reader may note that it is difficult to imagine a useful algorithm with an error that depends in this way on the problem being solved. Nevertheless, such an error function is possible, which is all that is required.

true because both X_n and Y_n are obtained by applying the same function to arguments which are identically distributed. Thus, although X_n and Y_n are not equal, depending as they do on different components of a sample point, $P\{ X_n \le x \} = P\{ Y_n \le x \}$. If one of the probabilities $P\{ |X_n| > \epsilon \}$ or $P\{ |Y_n| > \epsilon \}$ tends to zero, then so does the other one, since they are equal. This demonstrates part (b) of the theorem.

By far the most interesting point of the theorem is part (c), which shows the difference between the independent problem model and the incremental problem model. The implication is an easy consequence of the Borel-Cantelli Lemma and the fact that $X_n \sim Y_n$. Since $\{Y_n\}$ are independent, $\sum_n P\{ |Y_n| > \epsilon \}$ is finite for every $\epsilon > 0$. Therefore, $\sum_n P\{ |X_n| > \epsilon \}$ is finite for all $\epsilon > 0$, which means that $X_n \to_{as} 0$, even though $\{X_n\}$ are not independent.

This argument fails to prove the implication in the other direction, since the lack of independence of $\{X_n\}$ prevents us from concluding anything about the convergence of $\sum_n P\{ |X_n| > \epsilon \}$. However, a very instructive example shows that the reverse implication is sometimes false. This will be the key to exposing the underlying problem which has not been noticed so far in the literature on probabilistic approximation algorithms.

Suppose that $X_n(\vec{\omega}) = k(n)^{-1} \sum_{1 \le j \le k} Z_j(\vec{\omega})$, where $Z_j \sim \mathcal{R}(0, 1)$ is determined by ω_j, and $k(n)$ is non-decreasing and unbounded. Similarly, let $Y_n(\vec{\omega}) = k(n)^{-1} \sum_{1 \le j \le k} Z_{n,j}(\vec{\omega})$,

where $Z_{n,j} \sim \mathfrak{N}(0, 1)$ is determined by $\omega_{n,j}$.[6]

According to the strong law of large numbers (see, for example, Chung [1974], Theorem 5.4.2) the sequence $S_n = n^{-1} \sum_{1 \le j \le n} Z_j \to_{as} 0$. Since $X_n = S_{k(n)}$, by Lemma 2.5, $S_n \to_{as} 0$ implies that $X_n \to_{as} 0$. On the other hand, $\{Y_n\}$ are independent, with $Y_n \sim S_{k(n)}$. Lemma 2.5 says that $\{Y_n\}$ may not converge almost surely, which is in fact the case for this particular sequence. Because $Y_n \sim \mathfrak{N}(0, k(n)^{-1})$, we know that $\sum_n P\{ |Y_n| > \epsilon \}$ is finite for all $\epsilon > 0$ iff $k(n)$ grows faster than $\log n$. The choice $k(n) = \lfloor \log n \rfloor + 1$, for example, will result in $X_n \to_{as} 0$, but $\{Y_n\}$ will not converge almost surely. This example completes the proof of part (c). □

The proof of the last part of Theorem 2.8 makes an important observation about the strong law of large numbers. At first glance, it may seem slightly amazing that under the simple assumption that $\{Z_n\}$ are i.i.d. and have a finite mean μ, $S_n = k_n^{-1} \sum_{1 \le j \le k} Z_j \to_{as} \mu$ whenever $k(n)$ is non-decreasing and unbounded. Closer inspection reveals that because $\{S_n\}$ are partial sums, if the terms of a particular sequence $\{S_n(\vec{\omega})\}$ ever get close to zero, they are not likely to deviate greatly from zero thereafter. For the usual case $k(n) = n$, this is true because $S_n = (1 - n^{-1}) S_{n-1} + n^{-1} Z_n$, so that as $n \to \infty$, the contribution of the "incremental" summand Z_n gets rapidly smaller. In this sense, then, it is less of a surprise that $S_n \to_{as} 0$, because the random variables $\{S_n\}$ are defined in the incremental model.

---

6 . John Lehoczky suggested that the proof be simplified by letting Z_j and $Z_{n,j}$ be normally distributed.

An extension of the proof given here, using the central limit theorem, leads to a version of the strong law of large numbers for the independent problem model. The only difference is that two added conditions (that $Z_{n,j}$ have finite variance, and that k(n) grow faster than log n) are used to prove almost sure convergence of $\{k(n)^{-1} \sum_{1 \le j \le k} Z_{n,j}\}$.

2.3. History of Confusion

Theorem 2.8, part (c), explains the difference between the incremental problem model and the independent problem model: strong success, or generally almost sure convergence, in the independent model implies the same in the incremental model, but not vice versa. In this section, we will review the use of such phrases as "almost surely" and "almost everywhere" in the relevant papers, and show that there is considerable confusion not only between problem models, but even between almost sure convergence and convergence in probability.

The fact that the difference between the problem models has been overlooked by computer scientists who are applying probability theory is understandable, since it has apparently not been explicitly pointed out in these terms before. Nevertheless, the essence of the difference is recognized by standard probability theory texts in the framework of the Lindeberg-Feller version of the central limit theorem (see Chung [1974], pages 196-214), where "triangular arrays" of random variables are introduced. In weaker versions of the central limit theorem (Chung [1974], page 169) the partial sums S_n of the previous section are used exclusively. While this difference is noted in passing, it is not emphasized.

In a very difficult paper, Beardwood, Halton, and Hammersley [1959] prove that

> "the length of the shortest closed path through n points in a bounded plane region of area A is 'almost always' proportional to $(nA)^{-1/2}$ for large n." [7]

Their problem model is spelled out very precisely. An infinite sequence of points is chosen (according to $\vec{\omega}$) from a uniform distribution over the bounded region of area A, and $T_n(\vec{\omega})$ is the length of the shortest closed path (traveling salesman tour) through the first n points of the sequence. Their main result is that $(nA)^{-1/2}T_n \rightarrow_{as} \mu$ for some universal constant μ, which does not depend even on the shape of the region. In our terminology, the problems are generated incrementally, with the n^{th} problem differing from the $(n-1)^{st}$ problem only by the addition of one new point. Thus, the theorem states that $(nA)^{-1/2}T_n \rightarrow_{as} \mu$ in the incremental model. It is not known whether almost sure convergence holds in the independent model. This question is addressed further in the next section.

Borovkov [1962] gives what appears to be the first algorithmic use of stochastic convergence. Citing the results of Beardwood, et al., he claims that

> "for large n and certain bounds on F(x) (the distribution function of points) it is possible to determine a simple rule for construction of a (traveling salesman tour) ... such that (the ratio of its length to that of an optimal tour) 'almost always' does not exceed $1 + p$, $p > 0$." [8]

Borovkov's so-called "simple rule" actually requires the solution of a complex optimization problem. It is apparently intended to serve not as a feasible procedure for generating approximate tours, but as a means for proving an upper bound on the

---

7. Beardwood, Halton, and Hammersley [1959], page 299.

8. Borovkov [1962], page 1403 of the translation.

(unknown) constant μ. Also, his method gives $p = 0.48$ for points uniformly distributed in the unit square, so the relative error of such a tour does not converge to zero, but only remains less than 0.48, almost surely. Throughout his discussion of the traveling salesman and assignment problems he fails to define his problem model, although it is clear from other remarks that he does distinguish between convergence in probability and almost sure convergence.

Perepelica [1970], in a paper on finding Hamiltonian circuits, says that an algorithm "almost always arrives at an asymptotically precise solution" [9] whenever the relative error $X_n \to_{pr} 0$. His problem model is uncertain, but since he actually uses convergence in probability, it really does not matter which problem model is being considered (Theorem 2.8, part (b)).

Burtin [1974, 1975], writing about extremal properties of random graphs, gives an interesting interpretation of random graphs as evolving in time by the addition of new edges. Although this suggests the incremental model, the interpretation applies only to individual graphs and not to a sequence of them, leaving open the question of which model is being used. This is really a moot point, since he says that

> "a certain property of a random graph holds 'almost surely' if the probability that the random graph has this property tends to 1 as $n \to \infty$." [10]

He, like Perepelica, has apparently defined almost sure convergence as convergence in probability.

Like Beardwood, et al., Grimmett and McDiarmid [1976] give a very explicit

---

9. Perepelica [1970], page 1377 of the translation.

10. Burtin [1974], page 710.

definition of their problem model for coloring random graphs. They let a sample point $\vec{\omega}$ define an infinite random graph in which each potential edge exists with constant probability, independent of the presence of other edges. The n^{th} problem is determined by the subgraph induced by the first n vertices of the infinite graph, which is precisely the incremental model. They also clearly distinguish between almost sure convergence and convergence in probability.

In his Ph.D. thesis, DeWitt [1977] uses some of these results regarding random graphs to analyze optimization algorithms. He presents several theorems to the effect that certain relationships hold "almost surely", but uses the following definition of the term:

> "For many properties of random graphs, conclusions are made which are typically of the form: $G_{N,M}$ has property P almost everywhere (almost surely, almost always). This means that the probability that $G_{N,M}$ does not have the property converges to zero as N goes to infinity."[11]

Again, this is actually convergence in probability rather than almost sure convergence.

The most important paper in this area, from an algorithmic standpoint, is Karp [1976]. He gives a definition of a problem model and convergence which is based on the Borel-Cantelli Lemma, saying that a predicate X_n "holds almost everywhere" if $\sum_n P\{X_n \text{ does not hold}\} < \infty$. Unfortunately, it is not unmistakably clear from this definition which problem model he has in mind, since $\sum_n P\{X_n \text{ does not hold}\} < \infty$ implies that X_n holds almost surely in both the incremental and the independent problem models. He does, however, hint at the independent model when he continues:

---

11 . DeWitt [1977], page 13.

> "This is a very strong condition. It implies that, if we drew an infinite sequence of problem instances, one of each size, then, with probability one, the predicate X_n would be observed to fail only finitely often." [12]

In a later paper, Karp [1977] specifically adopts the independent model:

> "Suppose we form an infinite sequence $Z_1, Z_2, \ldots, Z_n, \ldots$ of independent samples ... ". [13]

Horowitz and Sahni [1977] have also interpreted Karp's original model to be the independent one, and Lewis and Papadimitriou [1978] imply, but do not actually state, the same interpretation of problem sequences. In his analysis of matching heuristics, Papadimitriou [1977b] is careful to explain the model used by Beardwood, et al., but does not mention the possibility of other models. The same remark applies to a recent paper by Stein [1978].

Assuming that Karp intended the independent model, as a subsequent paper seems to indicate, there are several misinterpretations in his 1976 paper. He relies on theorems proved by Beardwood, et al., and Grimmett and McDiarmid to the effect that certain sequences converge almost surely in the incremental model, even though Theorem 2.8 shows that almost sure convergence does not necessarily hold in the independent model. In his 1977 paper, he makes the same oversight, although it affects only the interpretation of what he has proved and not the proofs themselves.

It should also be noted that while Karp [1976] cites a result of Posa [1976] regarding a probabilistic approximation algorithm for finding Hamiltonian circuits, the latter claims to have proven only weak success of the algorithm. However, the proofs can easily be extended to demonstrate strong success in the independent model.

---

12 . Karp [1976], page 3.

13 . Karp [1977], page 216.

2.4. Strong Success in the Independent Model

There are three main reasons for wanting to deal with the independent model. The first, and most important with respect to interpretation of results, is that it "feels" more intuitively pleasing. The behavior of an algorithm on each problem is independent of its behavior on other problems. We can therefore use Karp's characterization of strong success based on $\sum_n P\{ |X_n| > \epsilon \}$, since the Borel-Cantelli Lemma says that, in the independent model, $X_n \rightarrow_{as} 0$ iff the sum is finite for all $\epsilon > 0$. Thus, strong success in this model means that if we independently chose one random problem instance of each size $n \geq 1$, then with probability one the algorithm would fail (that is, have error exceeding any fixed $\epsilon > 0$) on only a finite number of them.

Secondly, and most important with respect to proofs of stochastic success, we must simply determine $P\{ |X_n| > \epsilon \}$ in order to establish weak or strong success. There is no need to examine links between successive problems to accomplish this, so that a problem of size n can be considered alone. Proofs are simplified partly because the strategy for a proof is well understood beforehand, and should not rely too much on features of the problem class.

The proof of Beardwood, et al., that $(nA)^{-1/2}T_n \rightarrow_{as} \mu$ in the incremental model uses a common technique for proofs in that model, which Chung [1974] calls the "method of subsequences". In order to demonstrate that $X_n \rightarrow_{as} 0$, one shows that some subsequence $X_{n_m} \rightarrow_{as} 0$ as $m \rightarrow \infty$, then proves that for any $\vec{\omega}$ the other terms of the original sequence $\{X_n(\vec{\omega})\}$ cannot differ by much from the terms of the subsequence

$\{X_{n_m}(\vec{\omega})\}$. These facts together guarantee that the whole sequence converges whenever the subsequence converges, proving that $X_n \rightarrow_{as} 0$. This method is ideal for the incremental problem model, but is useless for the independent model where there is no relationship between successive terms of a sequence $\{X_n(\vec{\omega})\}$.

The final advantage of the independent model is that strong success implies strong success in the incremental model, and implies weak success in both models. It is, in effect, the "strongest" form of the four classes of stochastic success. Theorem 2.8 demonstrates that strong success in the independent model is strictly superior to the other possibilities in this sense.

2.5. Example: The Traveling Salesman Problem

Despite the advantages of the independent model, proofs can still be difficult, which will be painfully evident from the remainder of this section. We will concentrate on Karp's [1976] algorithm for the Euclidean traveling salesman problem. He originally claimed that the algorithm succeeds strongly in the independent model, but as we have seen, confusion between the models casts considerable doubt on this conclusion. Here we show that even if the almost sure convergence of normalized optimal tour length (proved by Beardwood, et al. [1959] for the incremental model) were also true in the independent model, as Karp assumed, his argument would still not prove the strong success of the algorithm in the independent model. In the process, we establish the groundwork for a proof of strong success of the algorithm in the independent model, and present heavy circumstantial evidence (but, unfortunately, not a complete proof) that the normalized optimal tour length converges almost surely in that model.

In a subsequent paper, Karp [1977] has given an extremely clever proof of the strong success of a closely related algorithm in the incremental model. This result would immediately be strengthened to hold in the independent problem model if the normalized optimal tour length were known to converge almost surely in the independent model. At the same time, however, he has implicitly rescinded his claim to strong success of the original algorithm, showing only that it gets within approximately a factor of two of optimal, almost surely, in the incremental model.

Before we examine Karp's probabilistic approximation algorithms in more detail, it is important to understand the results of Beardwood, et al. [1959] relating to the problem at hand. The Euclidean traveling salesman problem (ETSP) is defined by a set of n points in Euclidean space, in this case the plane. A solution consists of a closed path (called a tour) which passes exactly once through each point and has minimum length among all tours. Distances are Euclidean interpoint distances for the ETSP, although for a general TSP they may be arbitrary, reflecting travel costs, for instance.[14] Garey, Graham, and Johnson [1976] and Papadimitriou [1977a] have shown that the ETSP is NP-complete, and is thus a prime candidate for approximate solution by a probabilistic approximation algorithm.

As a simple probabilistic model, we imagine that the n points are chosen independently from a uniform distribution over the unit square. With T_n defined to be the length of the optimal tour through n such points, Beardwood, et al. show that $n^{-1/2}T_n \rightarrow_{as} \mu$, for some constant μ, in the incremental model. Most of their results, and ours, carry over with minor modifications to general distributions of points over

---

14. See Chapter 4 for more on the general problem.

Lebesgue-measureable regions in higher dimensions. We take only this special case because the proof is difficult enough and the issues are clear without tackling the more general problem.

Karp's probabilistic approximation algorithms are both based on partitioning the points into smaller problems which can be solved relatively quickly, then patching together solutions to the subproblems to produce a near-optimal tour through all n points. His original algorithm (Karp [1976]), which we call A_1, implements the partitioning by dividing the unit square into n/h(n) smaller congruent squares, each of which contains h(n) points on the average. Optimal tours are computed for the points within each subsquare, and these subtours are patched together by twice traversing each edge of a minimum spanning tree joining the subtours, as illustrated in Figure 2.2. The expected computation time required by A_1 is $O((n/h)c^h) + O(n \log^2 n)$ for some constant c.[15]

The second algorithm A_2 (Karp [1977]) partitions the points themselves rather than partitioning the region a priori. For each subset of points at any step, the points within it are split into two groups by a line segment joining the two longer sides of the region, which passes through the point with median coordinate in the same direction as those sides. This splits the region into two parts, as shown in Figure 2.3. The process is continued until there are no more than h(n) points in any subregion, whereupon optimal tours are computed for each group of points. These subtours are already

---

15. Karp [1977], page 219.

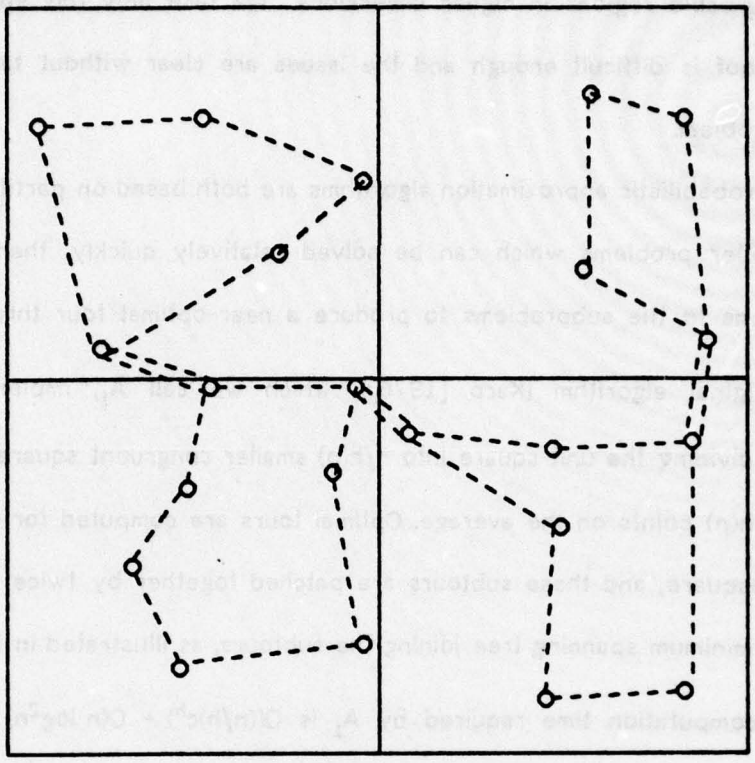


FIGURE 2.2 - Operation of Karp's original algorithm, A_1.

connected, so there is no need to find a minimum spanning tree as in the other algorithm. A_2 operates in time $O((n/h)c^h) + O(n \log n)$ for some constant c.[16]

It is intuitively clear that as a result of the triangle inequality, these closed paths can be transformed into tours (paths which visit each point exactly once) which are no longer than the original paths. The length of the original path therefore provides an upper bound on the length of the approximate tour finally produced. Karp [1977] shows precisely how the post-processing step can be accomplished.

Algorithm A_1, which will be considered in detail, works as follows:

---

16 . Karp [1977], page 212.

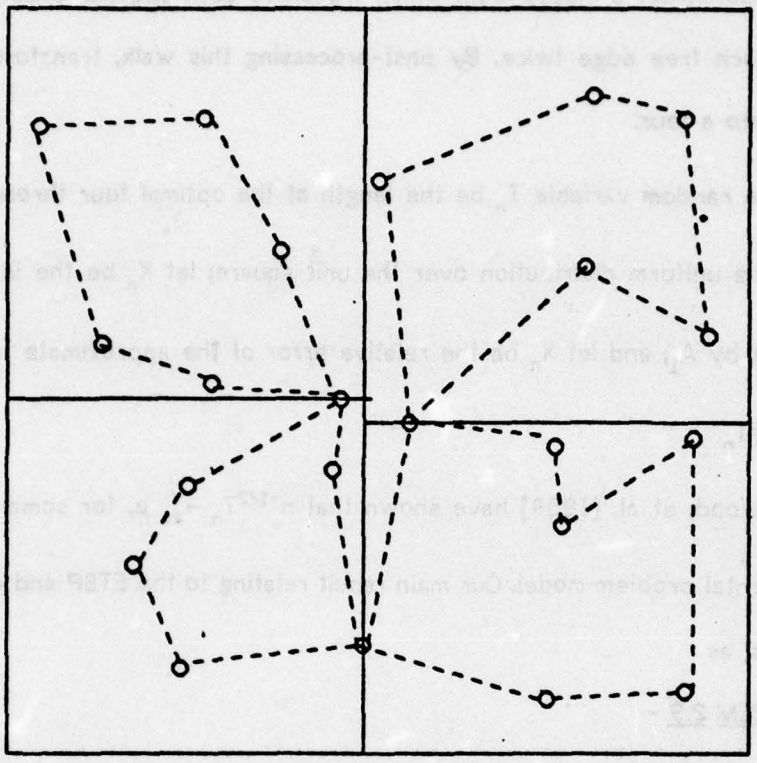


FIGURE 2.3 - Operation of Karp's second algorithm, A_2

(1) Let $h = h(n)$ be an non-decreasing unbounded function satisfying $h(n) = o(n)$ and n/h a perfect square. Partition the unit square into n/h smaller squares, each of area h/n.

(2) Find an exactly optimal tour through the points in each subsquare. Call these the subtours.

(3) Regarding each subtour as a point, with the distance between two subtours taken as the minimum distance between a point on one and a point on the other, find a minimum spanning tree joining the subtours.

(4) Construct a closed walk which traverses each subtour once and each tree edge twice. By post-processing this walk, transform it into a tour.

Let the random variable T_n be the length of the optimal tour through n points sampled from a uniform distribution over the unit square; let K_n be the length of the tour produced by A_1; and let X_n be the relative error of the approximate tour, so that $X_n = (K_n - T_n)/T_n$.

Beardwood, et al. [1959] have shown that $n^{-1/2}T_n \rightarrow_{as} \mu$, for some constant μ, in the incremental problem model. Our main result relating to the ETSP and A_1 is stated, in these terms, as

THEOREM 2.9 -

(a) There exist functions h(n), satisfying the conditions of A_1, such that for all $\epsilon > 0$, $n^{-1/2}K_n \leq \mu + \epsilon$, almost surely, in the independent model.

(b) If $n^{-1/2}T_n \rightarrow_{as} \mu$ in the independent model, then $X_n \rightarrow_{as} 0$ in that model. That is, there exist functions h(n) satisfying the conditions of A_1 such that A_1 succeeds strongly in the independent problem model.

Remark - Part (a) of this theorem constitutes some of the evidence that $n^{-1/2}T_n \rightarrow_{as} \mu$ in the independent model. It means that for every $\epsilon > 0$,

$P\{ \vec{\omega}: \limsup n^{-1/2}K_n(\vec{\omega}) \leq \mu + \epsilon \} = 1$ in that model. Since $T_n(\vec{\omega}) \leq K_n(\vec{\omega})$, this implies that $n^{-1/2}T_n \leq \mu + \epsilon$, almost surely, in the independent model, leaving only the establishment of an almost sure lower bound of $\mu - \epsilon$ as the obstacle to proving that $n^{-1/2}T_n \rightarrow_{as} \mu$ in the independent problem model.

Proof - The first major difficulty encountered in the proof is lack of independence of the numbers of points N_j in the subsquares. Since $\sum_{1 \leq j \leq n/h} N_j = n$, an excess of points in one subsquare tends to reduce the number in the others. This complicates the situation considerably, since we wish to explore the behavior of sums of subtour lengths, which are therefore also dependent.

This problem can be overcome by analyzing a modified algorithm which never produces a better approximate tour than the original algorithm A_1. The tour length from the modified version will still provide an upper bound on K_n, which is precisely what we want, but will allow us to define a set of subtour lengths which are i.i.d random variables. The central limit theorem will then permit calculation of the limiting distribution of the sum of subtour lengths, from which we can estimate the probabilities needed to prove the theorem.

Let $E_n = \max_j \{ |N_j/h - 1| \}$, and let $m = (1 - E_n)h$. If $E_n > 1$, then m is defined to be 0. Intuitively, E_n is the maximum absolute deviation from 1 of the ratio of the actual proportion of points in any subsquare, to the expected proportion of points there (h/n). Therefore, m is just small enough that each subsquare is guaranteed to contain

at least m points. The quantity m is a random variable, of course, as are subsequent quantities subscripted by m, but for notational convenience we depart from the upper-case convention here.

The modified algorithm does not find exactly optimal subtours, but only approximately optimal ones. We simply replace step (2) of A_1 by the following:

(2a) Find an exactly optimal tour through m randomly chosen points of each subsquare; call these the m-tours. Let $U_{n,j}$ be the length of the j^{th} m-tour.

(2b) For each subsquare, regarding the m-tour as a point, find a minimum spanning tree connecting it and the remaining points of the subsquare, if any.

(2c) Within each subsquare, construct a closed walk which traverses the m-tour once and each spanning tree edge twice. Post-process each walk (as suggested in step (4)) to produce a subtour through the points in each subsquare. (Note that this redefines subtours to be only approximately rather than exactly optimal.)

The remainder of the modified algorithm is the same as the original one. It is clear that the length of the tour produced by this new version cannot be shorter than that produced by A_1, since the subtours may not be optimal. The length of this tour is, however, no longer than the sum of the lengths of the m-tours, plus twice the sum of the lengths of the minimum spanning trees through the remaining (at most $2E_n h$) points in each subsquare, plus twice the length of the minimum spanning tree joining the subtours.

Beardwood, et al. [1959] have shown that for any connected planar region of area A, $n^{-1/2}T_n \leq CA^{1/2}$ for some constant C. Because the length of a minimum spanning tree through a set of points cannot exceed the length of the traveling salesman tour through those points, we know that the total length of the minimum spanning tree through the remaining points of a subsquare is bounded by $C(h/n)^{1/2}(2E_n h)^{1/2} = Ch(2E_n/n)^{1/2}$. Therefore, the contribution of these minimum spanning tree lengths to the final sum is at most $2C(2E_n n)^{1/2}$. Similarly, the length of the minimum spanning tree joining the n/h subtours is bounded by $C(n/h)^{1/2}$, so its contribution to the total is at most $2C(n/h)^{1/2}$. Therefore, the following inequalities are satisfied:

$$n^{-1/2}T_n \leq n^{-1/2}K_n \leq n^{-1/2}\sum_{1\leq j\leq n/h} U_{n,j} + 2C((2E_n)^{1/2} + h^{-1/2}).$$

The theorem will be proved if we can demonstrate that the expression on the right is at most $\mu + \epsilon$, almost surely, for any $\epsilon > 0$. In fact, we will show that in the independent problem model, the random variable represented by that expression converges almost surely to μ. This will prove part (b) of the theorem as well, for if $n^{-1/2}T_n \rightarrow_{as} \mu$ in the independent model, and the length of the tour produced by the modified algorithm converges almost surely to μ in that model, then $n^{-1/2}K_n \rightarrow_{as} \mu$. Applying the relative error lemma proves that A_1 succeeds almost surely. It is not known, however, whether $n^{-1/2}T_n \rightarrow_{as} \mu$ in the independent model.

Intuitively, we show that the relative contribution of all the minimum spanning tree lengths becomes negligibly small compared to the sum of the lengths of the m-

tours. This requires a proof that $E_n \rightarrow_{as} 0$, since $h^{-1/2} \rightarrow 0$ deterministically. We must show that

$$\sum_n P\{ E_n > \epsilon \} = \sum_n P\{ \max_j |N_j/h - 1| > \epsilon \}$$

is finite for all $\epsilon > 0$. This last expression is again difficult to evaluate because the N_j are not independent. However, if we consider a set of points generated by a Poisson process with rate parameter n (points per unit area), then the numbers of points M_j in each subsquare are independent, with

$$P\{ M_j = k \} = e^{-h} h^k / k!$$

(This is just the definition of a Poisson process with rate parameter n points per unit area, applied to an area of size h/n. The expression for the probability of k points in area h/n when the rate is n points per unit area is $e^{-n(h/n)} (n(h/n))^k / k!$ which is equal to the expression above.)

This formulation has the useful property that the conditional distribution of the points in the unit square, given that $\sum_{1 \le j \le n/h} M_j = n$, is the same as the distribution of n points from a uniform distribution over the same area. Therefore,

$$\begin{aligned} P\{ \max_j |N_j/h - 1| > \epsilon \} &= P\Big\{ \max_j |M_j/h - 1| > \epsilon \;\Big|\; \sum_{1 \le j \le n/h} M_j = n \Big\} \\ &\le P\{ \max_j |M_j/h - 1| > \epsilon \} \;/\; P\Big\{ \sum_{1 \le j \le n/h} M_j = n \Big\} \\ &= \Big(1 - P\{ \max_j |M_j/h - 1| \le \epsilon \}\Big) \;/\; P\Big\{ \sum_{1 \le j \le n/h} M_j = n \Big\} \\ &= \Big(1 - P\{ |M_j/h - 1| \le \epsilon,\ 1 \le j \le n/h \}\Big) \;/\; P\Big\{ \sum_{1 \le j \le n/h} M_j = n \Big\} \\ &= \Big(1 - \big(P\{ (1-\epsilon)h \le M_j \le (1+\epsilon)h \}\big)^{n/h}\Big) \;/\; P\Big\{ \sum_{1 \le j \le n/h} M_j = n \Big\} \end{aligned}$$

$$= \left(1 - \left(\sum_k e^{-h} h^k / k!\right)^{n/h}\right) e^n n! / n^n$$

where the sum $\sum_k$ is over $(1 - \epsilon) h \le k \le (1 + \epsilon) h$.

Using the normal approximation to the Poisson distribution as the rate parameter $n \to \infty$ and then estimating tail probabilities we have

$$P\{ \max_j |N_j/h - 1| > \epsilon \} = O((n/h)^{3/2} \exp(-\epsilon^2 h/2)) .$$

Applying the Borel-Cantelli Lemma shows that $E_n \to_{as} 0$ if h(n) grows faster than log n. This is the first condition that h should satisfy. Notice that we have proved only that this is sufficient for $E_n \to_{as} 0$, not that it is necessary.[17]

We must now show that $n^{-1/2} \sum_{1 \le j \le n/h} U_{n,j} \to_{as} \mu$. As mentioned before, $n^{-1/2} T_n$ is bounded, so that the quantities $\mu_n = E(n^{-1/2} T_n)$ and $\sigma_n^2 = D(n^{-1/2} T_n)$ exist, and furthermore, $\mu_n \to \mu$ and $\sigma_n^2 \to 0$. This follows because $n^{-1/2} T_n \to_{as} \mu$ in the incremental model, and therefore $n^{-1/2} T_n \to_{pr} \mu$ in the independent model (Theorem 2.8, parts (a) and (b)). The fact that $n^{-1/2} T_n$ is bounded guarantees the convergence of moments according to Lemma 2.4. Even though σ_n^2 may not be a strictly decreasing function of n, it is certainly bounded above and below by functions which decrease monotonically to 0. For simplicity, we will treat σ_n^2 itself as a decreasing function of n. A more formal argument would examine the bounding functions instead, but only at the cost of further

---

17 . On the other hand, we have been unable to prove a weaker sufficient condition on h(n).

complicating the proof. We opt for keeping things as simple as possible, with the proviso that a more rigorous treatment is possible.

Noting a simple scale change, it is clear that $U_{n,j} \sim (h/n)^{1/2} T_m$. Furthermore, $U_{n,j}$, for $1 \le j \le n/h$, are independent random variables. If we define $V_{n,j} = (n/h)^{1/2}\left((mh/n)^{-1/2} U_{n,j} - \mu_m\right) / \sigma_m$, we see that $E(V_{n,j}) = 0$ and $D(V_{n,j}) = h/n$.

Now we are ready to show that $n^{-1/2} \sum_{1 \le j \le n/h} U_{n,j} \to_{as} \mu$ by using the Borel-Cantelli Lemma. To do this, we must demonstrate the convergence, for all $\epsilon > 0$, of

$$\sum_n P\{ \, | n^{-1/2} \sum_{1 \le j \le n/h} U_{n,j} - \mu \, | > \epsilon \, \}.$$

$$\sum_n P\{ \, | n^{-1/2} \sum_{1 \le j \le n/h} U_{n,j} - \mu \, | > \epsilon \, \}$$

$$\le \sum_n P\{ \, | n^{-1/2} \sum_{1 \le j \le n/h} U_{n,j} - \mu_m (1 - E_n)^{1/2} \, | > \epsilon/2 \, \}$$

$$+ \sum_n P\{ \, | \mu_m (1 - E_n)^{1/2} - \mu \, | > \epsilon/2 \, \}$$

$$(*) \qquad = \sum_n P\{ \, | \sum_{1 \le j \le n/h} V_{n,j} \, | > \epsilon (n/h)^{1/2} / (2\sigma_m (1 - E_n)^{1/2}) \, \}$$

$$+ \sum_n P\{ \, | \mu_m (1 - E_n)^{1/2} - \mu \, | > \epsilon/2 \, \}$$

In order to establish the convergence of the first sum in (*) above, we divide the analysis into two mutually exclusive and exhaustive cases. For the first case, the Chebychev inequality is sufficient to prove convergence, while for the second case we need the central limit theorem.

Case 1 - There exist constants $c > 0$ and $\delta > 0$ such that $\sigma_n^2 \le c n^{-2-\delta}$. Then

$$P\{ \, | \sum_{1 \le j \le n/h} V_{n,j} \, | > \epsilon (n/h)^{1/2} / (2\sigma_m (1 - E_n)^{1/2}) \, \}$$

$$\le P\{ \, | \sum_{1 \le j \le n/h} V_{n,j} \, | > \epsilon (n/h)^{1/2} / (2\sigma_{h/2}) \, | \, \sigma_{h/2} \ge \sigma_m \, \}$$

$$\le P\{ \, | \sum_{1 \le j \le n/h} V_{n,j} \, | > \epsilon (n/h)^{1/2} / (2\sigma_{h/2}) \, \} \; / \; P\{ \, m \ge h/2 \, \}$$

$$\leq \left(4h\sigma_{h/2}^2 \ / (\epsilon^2 n) \right) \ / \ P\{ E_n \leq 1/2 \}.$$

For all sufficiently large n, $P\{ E_n \leq 1/2 \} \geq 1/2$, since $E_n \rightarrow_{as} 0$. So

$$P\left\{ \left| \sum_{1 \leq j \leq n/h} V_{n,j} \right| > \epsilon(n/h)^{1/2} / (2\sigma_m(1 - E_n)^{1/2}) \right\} \leq 8h\sigma_{h/2}^2 \ / (\epsilon^2 n)$$

for all sufficiently large n. Therefore, the first sum in (*) is finite for every $\epsilon > 0$ if $\sum_n 8h\sigma_{h/2}^2 \ / (\epsilon^2 n)$ is finite for every $\epsilon > 0$. This is indeed true, since

$$\sum_n 8h\sigma_{h/2}^2 \ / (\epsilon^2 n) \leq (2^{5+\delta} c \ / \ \epsilon^2) \sum_n 1 \ / (nh^{1+\delta})$$

which converges for any function h(n) satisfying the previous condition that h(n) grows faster than log n.

Case 2 - For every $c > 0$ and $\delta > 0$, $\sigma_n^2 > cn^{-2-\delta}$ for sufficiently large n.

Since $V_{n,j}$, for $1 \leq j \leq n/h$, are i.i.d. with $E(V_{n,j}) = 0$ and $D(V_{n,j}) = h/n$, we can apply the Lindeberg-Feller version of the central limit theorem, which says that if, for all $\tau > 0$,

$$(n/h) \int_{|x| > \tau} x^2 dF_n(x) \rightarrow 0$$

where $F_n(x)$ is the distribution function of $V_{n,j}$, then

$$\sum_{1 \leq j \leq n/h} V_{n,j} \rightarrow_d \mathfrak{N}(0, 1).$$

Since $n^{-1/2} T_n \leq C$, for some constant C, it is clear that $|V_{n,j}| \leq C(h/n)^{1/2}/\sigma_m \leq C(h/n)^{1/2}/\sigma_h$. By the hypothesis of case 2, then, $|V_{n,j}| \leq c'(h^{3+\delta}/n)^{1/2}$ for every $c' > 0$ and every $\delta > 0$, for sufficiently large n. This expression tends to 0 if $h(n) = o(n^{1/(3+\delta)})$

for some $\delta > 0$. For such functions h, $(n/h)\int_{|x|>\tau} x^2 dF_n(x) = 0$ for all sufficiently large n. As a result of the central limit theorem, we can conclude that

$$\sum_{1\le j\le n/h} V_{n,j} \to_d \mathfrak{N}(0,1)$$

or, in terms of $U_{n,j}$

$$(n/h)^{1/2}\left(n^{-1/2}\sum_{1\le j\le n/h} U_{n,j} - \mu_m(1-E_n)^{1/2}\right) \Big/ \left(\sigma_m(1-E_n)^{1/2}\right) \to_d \mathfrak{N}(0,1).$$

Estimating tail probabilities of the normal distribution, we find that

$$\sum_n P\left\{ \left| \sum_{1\le j\le n/h} V_{n,j} \right| > \epsilon(n/h)^{1/2} / (2\sigma_m(1 - E_n)^{1/2}) \right\}$$
$$= \sum_n O(\exp(-\epsilon^2 n / (8h\sigma_m^2(1 - E_n))) .$$

Since $\sigma_m^2(1 - E_n)$ is bounded, it is a simple matter to verify that the sum is finite for all $\epsilon > 0$ if $h(n) = o(n/\log n)$. This condition is weaker than the previous one. Our restriction is that for any function h(n) which grows faster than log n, but not as fast as $n^{1/(3+\delta)}$ for any $\delta > 0$, the first sum in (*) above is finite.

The theorem is therefore proved if we can show that the second sum in (*) also converges.

$$\sum_n P\{ |\mu_m(1 - E_n)^{1/2} - \mu| > \epsilon/2 \}$$
$$\le \sum_n P\{ |(1 - E_n)^{1/2} - 1| > \epsilon/(4\mu) \}$$
$$+ \sum_n P\{ |(1 - E_n)^{1/2}(\mu_m - \mu)| > \epsilon/4 \}$$
$$\le \sum_n P\{ |(1-E_n)^{1/2} - 1| > \epsilon/(4\mu) \} + \sum_n P\{ |\mu_m - \mu| > \epsilon/4 \} .$$

Note that again the first sum above converges, since $E_n \to_{as} 0$. Hence we have

reduced the problem to the convergence of the second sum. Since $\mu_n \to \mu$, there exists a constant $N = N(\epsilon)$ such that $n \geq N$ implies $|\mu_n - \mu| < \epsilon/4$. Therefore,

$$\begin{aligned}
&\sum_n P\{ |\mu_n - \mu| > \epsilon/4 \} \\
&\qquad \leq \sum_n P\{ m < N \} \\
&\qquad = \sum_n P\{ (1 - E_n)h < N \} \\
&\qquad = \sum_n P\{ E_n > 1 - N/h \}.
\end{aligned}$$

Since h(n) is grows without bound, this sum converges because $E_n \to_{as} 0$. This proves the theorem. We note that the result immediately generalizes to higher dimensions or to rectangles. For a k-dimensional hyper-rectangle of measure v, the appropriate normalizing factor for the tour length is $n^{-(k-1)/k} v^{-1/k}$. □

At first glance, Theorem 2.9 seems easy to prove. There are several possible approaches which look perfectly reasonable on the surface, but which overlook crucial details, and therefore lead to erroneous "proofs". The most important incorrect assumption is that $n^{-1/2} T_n \to_{as} \mu$ in the independent model, which remains to be proved. Even supposing this to be true, the road is filled with hazards.

For example, one might assume that each subsquare contains exactly h points. There is no doubt that this (over-)simplification makes the proof much shorter, but it ignores one of the main issues, which is now to deal with the fact that the actual numbers of points in the subsquares are not independent. Our proof accomplishes this by considering the maximum deviation from h/n of the fraction of points in any subsquare, and by considering the points to be generated by a Poisson process.

Avoiding this detail would allow one to incorrectly conclude that h(n) could grow arbitrarily slowly.[18] In fact, the condition that h(n) grow faster than log n means that the time required by the algorithm is not bounded by any polynomial in n, the problem size. Karp originally set h(n) = log log n so that the expected execution time would be good, without noticing that this might affect the stochastic success of the algorithm.

One is also tempted to combine this first assumption with another that makes the proof almost trivial. Since $U_{n,j} \sim (h/n)^{1/2} T_h$ under the assumption that all subtours traverse h points, and assuming that $n^{-1/2} T_n \rightarrow_{as} \mu$ in the independent model, it must certainly be true that $(h^2/n)^{-1/2} U_{n,j} \rightarrow_{as} \mu$. However, Lemma 2.5 shows that this does not necessarily follow. It would follow, according to Lemma 2.5, that $(h^2/n)^{-1/2} \rightarrow_{pr} \mu$, so the strong law of large numbers should certainly guarantee that $(h/n) \sum_j (h^2/n)^{-1/2} U_{n,j} = n^{-1/2} \sum_j U_{n,j} \rightarrow_{as} \mu$. This would be true in the incremental model, as the proof of Theorem 2.8, part (c), explains, but not necessarily in the independent model. Again, some growth condition on h(n) apparently must be imposed to assure almost sure convergence in the independent problem model.

2.6. Conclusions

This chapter has been a difficult one to write, and as the patient reader is well aware, even more difficult to read. However, it is important to the remainder of the thesis because it provides the foundation for a theory of probabilistic approximation

---

18 . As pointed out before, this conclusion may not be false, but it has certainly not been proved yet.

algorithms which is able to cope with some of the difficulties encountered in analysis of fairly complex algorithms. When a procedure is sophisticated enough to produce good approximations with high probability, it usually is complicated enough to resist the kind of analysis that would reveal the actual distribution of the answers it gives. Here we have seen how the concepts of stochastic convergence can help us evaluate probabilistic approximation algorithms even without distributional results.

The idea of retreating from the distribution of the answer to stochastic convergence is not new. However, we have seen how the modes of stochastic convergence have repeatedly been confused, and how the differences between the two problem models have been overlooked. The major contributions of this work include pointing out these difficulties and proposing a solution, showing how the proposal leads to new questions, and partially resolving one such question (the behavior of the ETSP in the independent problem model).

The reader should be happy to know that the remaining two chapters of this thesis require surprisingly little probability theory beyond that learned in a first course in the subject. Most of the material is based on simple probabilistic and statistical concepts, as opposed to the unfamiliar ideas of this chapter. The probabilistic approximation algorithms developed in Chapters 3 and 4 are shown to succeed strongly in the independent model under certain conditions, which is the primary reason for presenting the material of this chapter first. Fortunately, it is not necessary to know much more than the hierarchy of Theorem 2.8 to appreciate these results.

3. Randomization and Sampling

In a more pragmatic spirit than prevailed throughout Chapter 2, this chapter deals with several very simple principles from statistics which can be used to design and analyze algorithms with certain desirable properties. Specifically, these tools enable design of exact algorithms which have good expected running times for a wide range of input distributions, and probabilistic approximation algorithms which produce good approximations with high probability.

Some of the algorithms in this chapter are simplified versions of algorithms which have appeared previously, and can be analyzed in a simpler fashion than the original ones. Each of the probabilistic approximation algorithms is shown to succeed strongly in the independent problem model under certain conditions.

Sections 3.1 and 3.2 describe how statistically based algorithms can be classified by analogy with the classification of statistical procedures as non-parametric and parametric. In Section 3.3 we present several algorithm design principles which use statistical methods, and in Section 3.4 a number of examples of the application of the techniques to a wide variety of problems.

3.1. Classification of Algorithms

We define two types of algorithms which can be designed and analyzed using statistical principles. The first kind are randomized algorithms, a name introduced by Yao [1977]. A randomized algorithm, operating on a fixed input, may take any one of several possible computation paths, depending on some internally controlled random process. For instance, at certain points in the operation of the algorithm, a random number might be generated from some known distribution, and the next step of the algorithm determined by the outcome of this event. The computation path taken by such an algorithm is non-deterministic, so that even if it were run many times on the same input, it might never perform exactly the same steps. As a result, the running time of a randomized algorithm is a random variable, even for a fixed input. Carter and Wegman [1977] have used a similar technique which involves the random choice of an algorithm followed by application of that algorithm. This approach is, of course, a special case of a randomized algorithm.

The second type of algorithm is the probabilistic approximation algorithm discussed in Chapter 2. A probabilistic approximation algorithm finds a good approximate solution to a problem with high probability. Karp's [1976, 1977] algorithms for the Euclidean traveling salesman problem are examples of such algorithms.

As Figure 3.1 shows, the classes of randomized algorithms and probabilistic approximation algorithms are not disjoint. The theoretical model of probabilistic Turing machines proposed by Gill [1974] and the prime-testing algorithms of Rabin [1976] and Solovay and Strassen [1977], for example, fall in the intersection of the two subsets. Most probabilistic analyses have concentrated on algorithms outside both

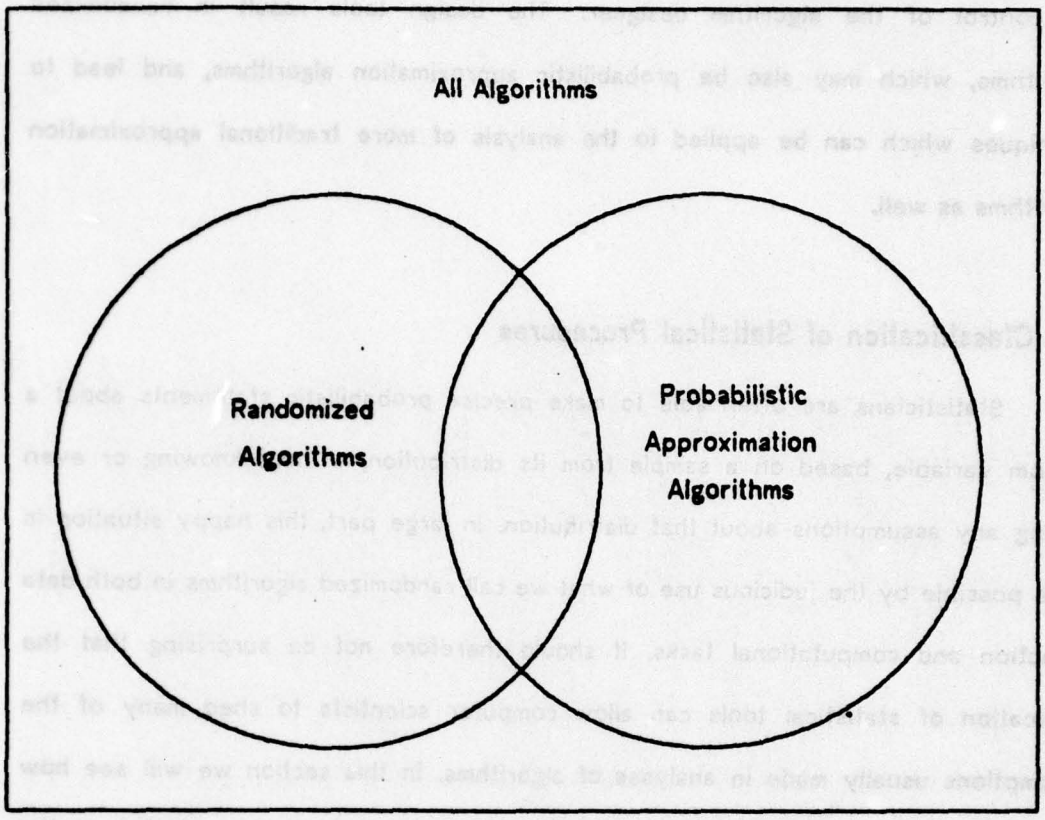


FIGURE 3.1 - A schematic representation of algorithm classes.

classes, with the probabilities introduced only to describe input characteristics. In such a model, the input to an algorithm is considered to be an observation on a random variable or a random structure. We will show how to extend many of these analyses, so that the results are valid with few or no restrictions on the input distribution, by applying probabilistic models to the flow of control of the algorithm (randomized algorithms) and to the outputs produced (probabilistic approximation algorithms). This will allow us to discard most assumptions about the unknown input distribution, which may often not be satisfied but which must be made in order to keep the mathematics tractable, in favor of assumptions about known distributions which are strictly under

the control of the algorithm designer. The design tools result in randomized algorithms, which may also be probabilistic approximation algorithms, and lead to techniques which can be applied to the analysis of more traditional approximation algorithms as well.

3.2. Classification of Statistical Procedures

Statisticians are often able to make precise probabilistic statements about a random variable, based on a sample from its distribution, without knowing or even making any assumptions about that distribution. In large part, this happy situation is made possible by the judicious use of what we call randomized algorithms in both data collection and computational tasks. It should therefore not be surprising that the application of statistical tools can allow computer scientists to shed many of the assumptions usually made in analyses of algorithms. In this section we will see how statisticians make use of random sampling methods, and in the remainder of the chapter see how the same ideas can be applied in the design and analysis of discrete algorithms.

The difference between randomized and probabilistic approximation algorithms corresponds to the (also fuzzy) distinction between non-parametric and parametric statistics. Non-parametric statistics deals primarily with the ranks of observations rather than with numerical values, and has the goal of making statements about the distribution of a function of a sample which does not depend on the underlying distribution of the observations (the population distribution). This essentially coincides with the goal of a randomized algorithm, which is to find the answer to a problem in expected time which is independent of the input distribution. On the other hand,

parametric statistics deals with the estimation of parameters of the population distribution. The distribution of such an estimate in general depends on population parameters, and may therefore require making a few assumptions about the underlying distribution. This estimate corresponds to the approximate solution produced by a probabilistic approximation algorithm, the analysis of which also usually requires a few weak assumptions about the input distribution.

A statistic is simply some function computed from a sample. As an example of the difference between the non-parametric and parametric varieties, suppose that we have a sample $S = \{x_i\}$ of n real numbers which are independent observations on a random variable X, and that we are interested in the population median (i.e., the median of the distribution of X, which is $\inf \{ x: F(x) \geq 1/2 \}$). A non-parametric approach is to guess that the median is M, and to compute the statistic g(S) which is the number of elements of S which are less than M. The question is then posed: "Suppose that M were the true population median, and that T were a random sample from the population. What is the probability that $| g(S) - n/2 | \leq | g(T) - n/2 |$?"

This question can be answered regardless of the distribution of X, since g(S) is known and $P\{ g(T) = k \} = \binom{n}{k} 2^{-n}$. The computed probability equals the probability that a truly random sample from a population with median M would give a value of the statistic g which was at least as far from n/2 as g(S). If this probability were very low, the statistician would reject the hypothesis that M is the median, since S is presumed to be a random sample. On the other hand, if the probability were high, he could not

reject the hypothesis; nor should he be convinced that M is the exact population median, since infinitely many choices for M could give exactly the same result! This somewhat circuitous approach is the price we pay for the ability to make distribution-free probability statements.

A parametric approach to the same problem is more straightforward. Rather than guessing the population median, we try to estimate it by, for example, finding the sample median (i.e., the median element of S). It is possible to describe accurately the form of the distribution of this statistic, and although the exact distribution depends on certain parameters of the (unknown) distribution of X, the form applies for a wide range of populations. The median problem will be further developed later in this chapter.

A key idea underlying both non-parametric and parametric statistics is the notion of sampling. In the median example, for instance, both approaches rely on the assumption that the sample S is a set of independent observations on a random variable X with some unknown distribution, or equivalently, an observation on a random vector in which each coordinate has the same (unknown) distribution.

A similar but much more restrictive assumption is usually made in probabilistic analyses of discrete algorithms, where the input is assumed to be an observation on a random structure with some specific distribution. For example, the typical expected-time analysis of a sorting algorithm assumes that the input (a permutation of n items) is equally likely to be any of the n! possible permutations. It is this added assumption about the input distribution which severely limits the apparent utility of probabilistic results, and which can often be eliminated if statistical techniques are used.

3.3. Design Principles for Randomized Algorithms

The design of a randomized algorithm simply involves the introduction of a randomization or sampling step into some stage of a deterministic algorithm. To statisticians, this step is part of the data collection process, and is assumed to have been handled properly. If we wish to place few or no limitations on the distribution of inputs to an algorithm, however, we cannot make such an assumption, which is tantamount to the assumption of equally likely or uniformly distributed inputs.

Several techniques can be shown to be effective in overcoming such assumptions. Some of them consider the input as being fixed, and make no probabilistic assumptions whatsoever. Others assume only that the input to an algorithm is an observation on a random structure, and by statistical means attempt to characterize certain features of the underlying distribution of inputs. The algorithm is then tailored so that it should work well for inputs from a distribution having such characteristics.

This type of algorithm is really an adaptive algorithm. Another approach to adaptive algorithms is presented by Holland [1975], who points out the similarities between natural and artificial adaptive systems and develops a formal model for studying these relationships. His very interesting "genetic adaptive algorithms" are investigated under this model, and in general cannot be easily analyzed for expected execution time. The adaptive algorithms presented here, on the other hand, are much simpler but can be analyzed. It remains an open research problem to integrate the two approaches and to exploit the advantages of each.

3.3.1. Randomization

One very simple approach to designing a randomized algorithm was suggested by Yuval [1975b] for the sorting problem. Expected-time analysis of a sorting algorithm, such as Quicksort, is usually prefaced with the assumption that each possible input permutation is equally likely (see Knuth [1973], Sedgewick [1975]). The reason for this is that the mathematics is particularly tractable, and elegant, if the partitioning element is equally likely to have each of the possible ranks 1 through n, and this is true if all possible permutations are equally likely. Yuval proposed that the input sequence simply be randomly permuted prior to application of the original Quicksort algorithm. This produces a randomized algorithm which runs in time O(n log n) on the average, regardless of the input distribution! Yuval's result and the technique are generalized in the following lemmas.

LEMMA 3.1 -

> For every fixed problem instance of size n, let T(n) be an upper bound on the expected time required by a randomized algorithm, averaged over all possible computation paths which it might follow. Then T(n) is an upper bound on the expected computation time, averaged over any distribution of inputs of size n.

Proof - Let $R_{i,n}$ be the expected computation time, averaged over all possible computation paths, for problem i of size n, and let R_n be the expected computation time averaged over a distribution of problems of size n with P{ problem i is the input } = p_i. Then we have, by definition,

$$R_n = \sum_i R_{i,n}\, p_i$$

$$\leq \sum_i T(n)\, p_i$$

$$= T(n)\sum_i p_i$$

$$= T(n)$$

which proves the lemma. □

LEMMA 3.2 - (Principle of randomization) -

Let A be a deterministic algorithm which takes as input a sequence of n items, and which runs in expected time T(n) if all possible permutations of the input sequence are equally likely. Let B be the algorithm which consists of randomly permuting the input sequence, then applying algorithm A to the result. Algorithm B runs in expected time T(n) + O(n) for any distribution of input permutations.

Proof - For any fixed input sequence of size n, B runs in expected time which is the sum of the time required to "shuffle" the input in such a way that each resulting permutation is equally likely, plus the time to solve the problem using algorithm A given that each input sequence is equally likely. The latter contribution is T(n) by assumption, so it only remains to be shown that the shuffling operation can be done in O(n) expected time, after which application of Lemma 3.1 proves the result.

The procedure "shuffle" below accomplishes this. It makes use of an auxiliary procedure "randomint(a,b)" which returns an integer equally likely to be any integer between a and b inclusive. This, in turn, uses "uniform(a,b)", which returns a real number uniformly distributed over [a,b). We assume a model of computation in which randomint(a,b) can be calculated in constant time, which is true for current digital computers.

```
integer procedure randomint(a,b);
   integer a,b;
   randomint := ⌊uniform(a,b+1)⌋;
```

The arguments to the random permutation procedure are an array to be permuted, X, along with lower and upper bounds on its indices, lb and ub, respectively. The construct "a :=: b" means that the contents of a and b are exchanged. Note that this procedure could be used to produce random permutations of the integers 1 through n by initializing X so that X[i] = i, for example, and then calling shuffle(X,1,n).

```
procedure shuffle(X,lb,ub);
   array X; integer lb, ub;
   begin
      integer i;
      for i := lb to ub-1 do X[i] :=: X[randomint(i,ub)]
   end;
```

The procedure obviously operates in time O(n), where n = ub - lb + 1, and its correctness is easily verified by using induction on n. See Knuth [1969] for more about shuffling algorithms. □

The principle of randomization, Lemma 3.2, gives a technique which applies to a wide variety of problems and, as we will see especially in Section 3.4.4, its ramifications are sometimes more subtle than they appear at first. Nevertheless, it does seem that there may often be easier ways to achieve the desired result than the rather drastic step of complete randomization. For instance, all that is really required in the analysis of Quicksort is that the partitioning element be equally likely to have each of the ranks 1 through n. This suggests that the random step be taken when the partitioning element is chosen, for example by partitioning around X[randomint(lb,ub)].[1]

---

1. Hoare himself proposed just this solution to the problem of assuming equally likely input permutations. See his original paper on Quicksort (Hoare [1962]).

Furthermore, it seems a shame to intentionally randomize an input file which might already be essentially sorted, just so that our analysis applies! On the other hand, even if an evil adversary tried to present the algorithm with a worst-case input sequence, he could not hope to require the algorithm to use more than O(n log n) expected time.

3.3.2. Approximation in Rank

The above objections to outright randomization as a preprocessing step highlight the differences between this brute force approach and a more practical but still very simple technique, namely sampling. We identify several distinct types of sampling which lead to effective randomized algorithms, both exact algorithms and probabilistic approximation algorithms. The first of these we call approximation in rank.

Suppose, for example, that we are faced with a discrete minimization problem over some feasible set. The basic idea of this approach is to pick at random some large number m of feasible solutions (if this is easily done) and to approximate the optimal solution by the one of these m which has the minimum objective function value. We intuitively believe that such a solution is likely to be close to the optimal, if only in the sense that about 1/m of all feasible solutions should be better. The "λ-opt heuristic" popularized by Lin [1965] is an example of this technique (see also Section 3.4.3).

The following lemmas generalize this result. Before presenting them, we introduce the beta distribution function $B_{k,h}(x)$, which is defined as follows:

$$B_{k,h}(x) = I_{k,h}(x) / I_{k,h}(1) \qquad 0 \le x \le 1$$

where $I_{k,h}(x) = \int_0^x t^{k-1}(1-t)^{h-1}dt$; we also have $B_{k,h}(x) = 0$ for $x < 0$ and $B_{k,h}(x) = 1$ for $x > 1$. We say that the random variable X has a beta distribution with parameters k and h, written $X \sim \beta(k,h)$, if $P\{ X \le x \} = B_{k,h}(x)$.

The notation for dealing with totally ordered sets is that used by Floyd and Rivest [1975], where $k \theta S$ is the k^{th} smallest element of set S, and $t \rho S$ is the rank of the element t in S (i.e., the number of elements of S which are less than or equal to t); hence, $(k \theta S) \rho S = k$.

LEMMA 3.3 - (Sampling with replacement)

Given a totally ordered set N of n elements, and a multiset M of m elements, each of which is independently randomly chosen from N (so that M is a random sample drawn with replacement from N). Let $X_k = (k \theta M) \rho N$, *and let* $Y_k = X_k/(n+1)$. *Then:*

(a) $p_j = P\{ X_k = j \} = \binom{j-1}{k-1}\binom{n-j+m-k}{m-k} / \binom{n+m-1}{m}$

$= \binom{-k}{j-1}\binom{-m+k-1}{n-j} / \binom{-m-1}{n-1}$

(b) $E(X_k) = 1 + k(n-1)/(m+1)$

(c) $D(X_k) = k(m-k+1)(n-1)(n+m) / \left((m+1)^2(m+2)\right)$

(d) *If* $m = o(n^{1/2})$ *then, as* $n \to \infty$, $Y_k \to_d \beta(k,m-k+1)$.

(e) *If* $m \to \infty$ *with* $m = o(n^{1/2})$, *and if* $k/(m+1) \to \alpha$ *for some constant* $0 < \alpha < 1$, *then as* $n \to \infty$

$$(Y_k - E(Y_k)) / (D(Y_k))^{1/2} \to_d \mathfrak{N}(0, 1)$$

Proof - Let $t = k \theta M$, so that $X_k = t \rho N$. For part (a), note that $X_k = j$ (i.e., $(k \theta M) \rho N = j$, or $k \theta M = j \theta N$) iff M contains exactly k-1 of the j elements less than or equal to t, contains t itself, and contains exactly m-k of the n-j+1 elements greater than or equal to t. There are $\binom{j+k-2}{k-1}$ ways of choosing the smaller elements (because we are sampling with replacement) and, for each of these, $\binom{n-j+m-k}{m-k}$ ways of choosing the larger ones. The total number of possible samples M is $\binom{n+m-1}{m}$, each of which is equally likely. This proves the first expression for p_j from which the second expression is obtained by applying the relation $\binom{n}{m} = (-1)^{n-m}\binom{-m-1}{n-m}$. Parts (b) and (c) involve computing the sums $\sum_j jp_j$ and $\sum_j j^2p_j$ which are done easily using the second form of p_j. The asymptotic approximations of parts (d) and (e) are obtained by using an expansion for the gamma function (see Abramowitz and Stegun [1965]). The detailed proofs are too long to include here. □

LEMMA 3.4 - (Sampling without replacement)

Given a totally ordered set N of n elements, and a randomly chosen subset $M \subseteq N$ of m elements (so that M is a random sample drawn without replacement from N), let $X_k = (k \theta M) \rho N$, and let $Y_k = X_k/(n+1)$. Then:

(a) $p_j = P\{ X_k = j \} = \binom{j-1}{k-1}\binom{n-j}{m-k}/\binom{n}{m}$

(b) $E(X_k) = k (n+1) / (m+1)$

(c) $D(X_k) = k (m-k+1) (n-m) (n+1) / \left((m+1)^2(m+2)\right)$

(d) If $m = o(n^{1/2})$ then, as $n \to \infty$, $Y_k \to_d \beta(k,m-k+1)$.

(e) If $m \to \infty$ with $m = o(n^{1/2})$, and if $k/(m+1) \to \alpha$ for some constant $0 < \alpha < 1$, then as $n \to \infty$

$$(Y_k - E(Y_k)) / (D(Y_k))^{1/2} \to_d \mathfrak{N}(0, 1).$$

Proof - Let $t = k \theta M$, so that $X_k = t \rho N$. For part (a), note that $X_k = j$ (i.e., $(k \theta M) \rho N = j$, or $k \theta M = j \theta N$) iff M contains exactly k-1 of the j-1 elements smaller than t, contains t itself, and contains exactly m-k of the n-j elements larger than t. This happens for precisely $\binom{j-1}{k-1}\binom{n-j}{m-k}$ choices of M, for which there are $\binom{n}{m}$ equally likely possibilities. Proofs of parts (b) and (c) involve computing the sums $\sum_j jp_j$ and $\sum_j j^2 p_j$, which can be done by standard methods. Proofs of parts (d) and (e) are similar to those of Lemma 3.3. □

Among the many special consequences of Lemmas 3.3 and 3.4 is the intuitively obvious asymptotic equivalence of sampling with and without replacement, under certain conditions on the sample size. Furthermore, for fixed relative rank $\alpha = k/(m+1)$ of an item in the sample, the probability that Y_k will differ by much from its expected value becomes very small as $n, m \to \infty$. Under the conditions of part (e) of the lemmas,

$$P\{ |Y_k - E(Y_k)| > \epsilon \} = P\{ |Y_k - E(Y_k)| / (D(Y_k))^{1/2} > \epsilon / (D(Y_k))^{1/2} \}$$

$$\to \left(2D(Y_k)/(\pi\epsilon^2)\right)^{1/2} \exp\left(-\epsilon^2/(2D(Y_k))\right).$$

Since we have $D(Y_k) \to \alpha(1-\alpha)/m$,

$$P\{ |Y_k - E(Y_k)| > \epsilon \} \to \left(2\alpha(1-\alpha)/(\pi\epsilon^2 m)\right)^{1/2} \exp\left(-\epsilon^2 m/(2\alpha(1-\alpha))\right)$$

which, for fixed $\epsilon > 0$, tends rapidly to 0 as m increases. For example, with $m = 999$, $P\{ |Y_{500} - 1/2| > .05 \} < .01$. This result immediately suggests approximation algorithms for selection problems (see Section 3.4.2).

3.3.3. Estimation and Subsampling

Finding a solution which has rank near that of the exact answer is sometimes the best we can hope for, since there may be no reasonable way to measure distance between solutions. For example, suppose we have the following list of names and are asked to find the median name in alphabetical order.

Bentley
Eddy
Jones
Kadane
Kung
Shamos
Smith

If an approximation algorithm produced Kung as the answer, we would probably say that the approximation was close, not because |Kung - Kadane| is small (no such distance measure is defined in the problem), but because |rank(Kung) - rank(Kadane)| = 1 is small compared to 7, the total number of names. On the other hand, if the list consisted of real numbers rather than names, the actual difference between the numbers themselves would be naturally defined and might be more useful.

There are several differences between the problem defined on abstract totally ordered sets and that defined on a set of real numbers. The first one has already been identified to be the availability of a distance measure when we have real

numbers. Another is the ability to "interpolate" (i.e., to produce new elements of the set from the given ones), which opens up the possibility of an approximate answer not being one of the original problem elements. A third difference is that there are infinitely many real numbers, a point which gets to the heart of an issue which will be addressed in Chapter 4.

The conclusion to be drawn from this discussion is that approximation in rank is a legitimate approach in the general setting, but when we have real numbers defining our problem we might be able to use more powerful techniques which take advantage of these special features. We will call this method approximation in value, or estimation.

For the statistician, the problem of estimation is to approximate the value of some parameter of the population distribution (given a sample) by computing a statistic, which is some function of the sample. For the computer scientist, the problem is the exact computation of that function. The statistician's estimate is presumably close to the true value of the parameter, but it is still only a probabilistic approximation, in the sense that the estimate is not necessarily equal to the exact value of the parameter. The computer scientist's exact answer, on the other hand, is precisely what the statistician wants to compute.

These complementary roles suggest a general approximation method for the computer scientist. If the exact computation of the required function would consume too many resources, he could simply estimate the appropriate population parameter in some other less costly (and perhaps less accurate) way, and use that estimate as an approximation to his exact answer. This works because both estimates should be close to the true value of the same population parameter and, therefore, close to each other. Furthermore, only weak distributional assumptions (if any) about the input should be

necessary, since the value of the population parameter need not be known in advance in order to carry out this procedure.

Here we introduce a general technique for probabilistic approximation which does not require that a completely new estimate for the population parameter be found. We call this method subsampling. Assume that a problem instance is specified by a set S of n real numbers chosen independently from the same (possibly unknown) distribution F(x), and a function g(S) which is the value of the problem solution. The basic idea is to average the values of problem solutions for many smaller problems, and to scale this average in such a way that it provides a good approximation to the answer to the single large problem. Keep in mind that the following lemma gives only a general description of subsampling, and that more specific results may be obtained for certain problems.

LEMMA 3.5 - (Subsampling)

Let $X_n = g(S)$ be the solution value for the input set S whose n elements are independently chosen from F(x); let T(n) be an upper bound on the time to compute g(S); and suppose that there is some class A of distribution functions such that for each $F \in A$, there exists a function h(n) and a constant $c \neq 0$ such that $X_n h(n) \rightarrow_{pr} c$. Let S be randomly partitioned into n/r subsets R_i each of size r; let $Y_{n,i} = g(R_i)$ for $1 \leq i \leq n/r$; and let $Z_n = (rh(r)/(nh(n)))\sum_i Y_{n,i}$. If $F \in A$ and $r(n) \rightarrow \infty$ as $n \rightarrow \infty$, then $(X_n - Z_n) / X_n \rightarrow_{pr} 0$. Furthermore, Z_n can be computed in $(n/r)T(r) + O(n)$ time.

Proof - Assume that $F \in A$ and that $r(n) \to \infty$. We first have $Y_{n,r}h(r) \to_{pr} c$ by Lemma 2.5(b), and it is clear that $Z_n h(n) \to_{pr} c$ since the average of i.i.d. random variables, each of which converges in probability to c, must also converge in probability to c. Application of the relative error lemma (2.7(a)) shows that $(X_n - Z_n) / X_n \to_{pr} 0$. Knowing h, one can compute Z_n by first randomizing S in O(n) time, then computing $g(R_i)$ for $1 \le i \le n/r$ in at most $(n/r)T(r)$ time, and finally averaging these solution values and scaling the result in $O(n/r) = O(n)$ time. □

There are several important facts to note about this lemma. First of all, it does not simply say that the expected value of the approximation approaches that of the true answer, nor that the expected value of the relative error approaches 0. Rather, it says something about the asymptotic distribution of the relative error. In most cases of interest, such as the case where $X_n h(n)$ is bounded, one can prove the convergence of expected values as a special consequence of this result.

Notice also that even though the most significant computational task (computation of the solutions to the smaller problems) can make use of the same algorithm which would ordinarily be used to solve the large problem, it is by no means obvious what the function h is. For certain kinds of problems, this is known. For the traveling salesman problem and the Steiner tree problem defined on points drawn independently from any distribution over a Lebesgue-measurable set in k-dimensional Euclidean space, $h(n) = n^{-(k-1)/k}$, as we saw in Chapter 2. While it is true that the constant c is not known in this case, it is not required for the computation of Z_n.

We can get an idea of the time savings possible by using subsampling by

considering a problem whose exact solution has $T(n) = 2^n$. Letting $r(n) = \log n$ results in an approximation algorithm that runs in time $O(n^2/\log n)$ and succeeds weakly, using the terminology of Chapter 2. In fact, it is possible to choose r to get a probabilistic approximation algorithm which runs in time $O(nf(n))$ for any function $f(n)$ which tends to infinity, but Lemma 3.5 does not provide a means for determining how good its approximations are likely to be, except to say that the relative error converges in probability to 0.

In Section 3.4.2, we will see how subsampling can be used to save space as well as time for a selection problem. For this problem, enough is known about the distribution of solution values that we can bound the probability that the relative error exceeds ϵ.

3.3.4. Empirical Distribution Functions

Many algorithms have been analyzed under the assumption that the inputs are real numbers which are uniformly distributed between 0 and 1 (written U(0,1)). Among these are algorithms for the important problems on totally ordered sets restricted to sets of real numbers, including sorting and searching (see Knuth [1973] and Dobosiewicz [1978]). A widely known but often overlooked method is available for extending these results to arbitrary distributions, and is presented as

LEMMA 3.6 -

> Let the random variable X have the distribution function F(x). Define the random variable Y so that $Y = F(X)$ if F is continuous at X, and $Y \sim U(F(X^-),F(X))$ if F has a jump discontinuity at X. Then $Y \sim U(0,1)$.

Proof - Let y be fixed, $0 \le y \le 1$, and let $x = F^{-1}(y) = \inf\{ x: F(x) \ge y \}$. If F is continuous at x, so that $y = F(x)$, then

$$\begin{aligned} P\{ Y \le y \} &= P\{ F(X) \le y \} \\ &= P\{ X \le F^{-1}(y) \} \\ &= F(F^{-1}(y)) \\ &= y. \end{aligned}$$

Similarly, if F has a jump discontinuity at x, so that $F(x^-) < y \le F(x)$, then

$$\begin{aligned} P\{ Y \le y \} &= P\{ Y \le y \mid X \le x^- \} \cdot P\{ X \le x^- \} \\ &\quad + P\{ Y \le y \mid x^- < X \le x \} \cdot P\{ x^- < X \le x \} \\ &= F(x^-) + ((y - F(x^-)) / (F(x) - F(x^-))) \cdot (F(x) - F(x^-)) \\ &= y \end{aligned}$$

which means that Y has the uniform distribution U(0,1). □

Intuitively, the lemma says that for a random variable X having a continuous distribution F, the random variable F(X) is uniformly distributed. The importance of this result for problems on totally ordered sets rests with the fact that the transformation F is increasing, and therefore preserves the ordering of real numbers. Thus, if we wish to sort a set $\{X_i\}$ of real numbers each having the distribution F, and we have an algorithm which sorts uniformly distributed real numbers, we can simply use that algorithm to sort $\{F(X_i)\}$, assuming F is continuous. The analysis of the algorithm for uniformly distributed inputs will still apply, since $F(X_i) \sim U(0,1)$. Furthermore, the set $\{X_i\}$ will be sorted when the set $\{F(X_i)\}$ is sorted because $F(X_i) \le F(X_j)$ iff $X_i \le X_j$. The

minor modification which seems to be required if F is not continuous may not really be necessary if our sorting algorithm is sophisticated enough to recognize equal items as a special case. In any event, it is easy enough to implement the transformation in constant time per item, adding only O(n) steps to the time used by the algorithm.

This approach is fine if F is known and if we can compute F(x). In many practical circumstances, however, F is not given as part of the problem. One obvious solution, of course, is to try to estimate F by sampling the inputs, and to produce a function called the empirical cumulative distribution function (ECDF) which is close to F. Given a random sample $\{X_i\}$ of size m from a distribution F(x), let $X_{(j)}$ be the j^{th}-smallest sample value. We define the ECDF $F_m(x)$ as

$$F_m(x) = 0 \qquad x < X_{(1)}$$

$$= (j-1+(x - X_{(j)})/(X_{(j+1)} - X_{(j)})) / (m-1) \qquad X_{(j)} \le x \le X_{(j+1)}$$

$$= 1 \qquad X_{(m)} \le x$$

This definition of the ECDF is not standard, but has special properties which will be of particular interest to us. The usual ECDF, call it $G_m(x)$, has $G_m(x) = j/m$ for $X_{(j)} \le x < X_{(j+1)}$ and $1 \le j < m$. Figure 3.2 shows what F_m and G_m might look like for $m = 4$. Notice that F_m is continuous, whereas G_m is not.

The next lemma illustrates that for large sample sizes, F_m is very likely to be close to the true distribution function F.

LEMMA 3.7 - (Glivenko-Cantelli Theorem)

As $m \to \infty$, $\sup_x | F_m(x) - F(x) | \to_{as} 0$ in the incremental model.

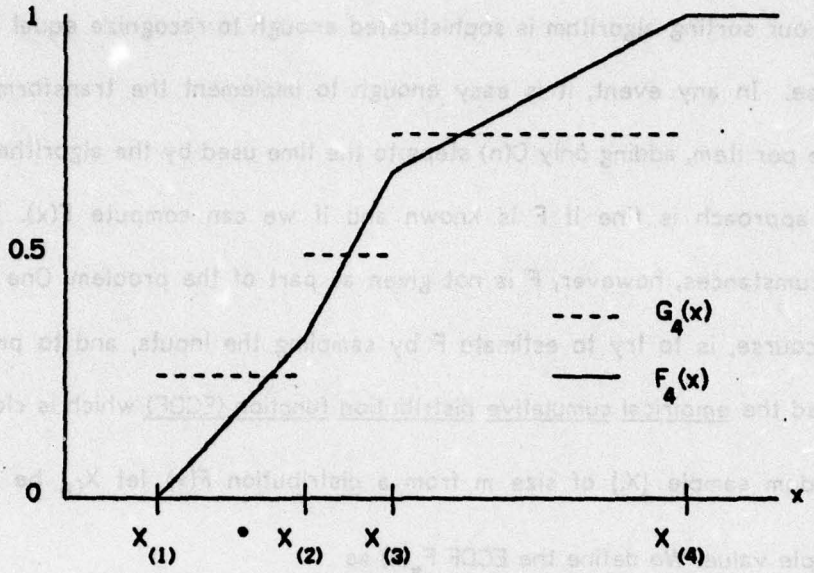


FIGURE 3.2 - The empirical cumulative distribution function.

Proof - See Chung [1974], page 133, for a proof with G_m replacing F_m. The lemma follows from this and the fact that $| F_m(x) - G_m(x) | \leq 1/m \rightarrow 0$. □

For every $\epsilon > 0$, then, we can expect F_m to be within ϵ of F with probability arbitrarily close to one, if the sample size m is sufficiently large. Unfortunately, while this is of interest in theory, it is not quite as strong a condition as we will need to extend analyses of algorithms from the case of uniformly distributed inputs to inputs from arbitrary distributions. The condition we have now guarantees that for any $\epsilon > 0$, there is some finite sample size m such that, for any $a \leq b$, $F(b) - F(a) \leq F_m(b) - F_m(a) + \epsilon$, almost surely. The condition which hindsight will show is more useful (after Section 3.4.1) is that there exists some constant c for which $F(b) - F(a) \leq$

$c(F_m(b) - F_m(a))$. In order to prove such a result, it is necessary to introduce some more terminology. We say that the function F satisfies a Lipschitz condition[2] iff there exists a constant C such that for all x and y, $|F(x) - F(y)| \leq C|x-y|$. Notice that if F has a finite derivative everywhere, then it satisfies a Lipschitz condition. On the other hand, no distribution function with a discrete or singular continuous component can satisfy a Lipschitz condition. The condition is therefore equivalent to requiring that F have a density which is everywhere bounded.

LEMMA 3.8 -

Let $\{X_i\}$ be a sample of size m from the distribution F(x), and let $F_m(x)$ be the ECDF as defined above. If F satisfies a Lipschitz condition with constant C, then for every a and b satisfying

$X_{(1)} \leq a \leq b \leq X_{(m)}$,

$$F(b) - F(a) \leq C\,(m-1)\,(X_{(m)} - X_{(1)})\,(F_m(b) - F_m(a))$$

Proof - By hypothesis, the constant C satisfies $|F(b) - F(a)| \leq C|b-a|$ for every a and b. We also have $F_m(b) - F_m(a) \geq s_m\,(b-a)$ for $X_{(1)} \leq a \leq b \leq X_{(m)}$, where $s_m = \min\{\,1 / ((X_{(j+1)} - X_{(j)})(m-1))\,\}$ is the minimum slope of F_m in the desired range. Using these two relations gives

$$F(b) - F(a) \leq (C/s_m)\,(F_m(b) - F_m(a))$$

which proves the lemma, since $s_m \geq 1/((m-1)(X_{(m)} - X_{(1)}))$. □

---

2. This is usually called a Lipschitz condition of order one, but since no other order of Lipschitz condition will be mentioned here, for brevity the order will be omitted and be assumed to be one.

Lemma 3.8 is quite different than Lemma 3.7 because it holds for any fixed value of m, not just as $m \to \infty$. What we are really doing by the piecewise linear function is approximating F by a distribution which is uniform over the m-1 intervals comprising the desired range of values. For m = 2, a very important special case, this amounts to approximating F by a uniform distribution over the range of interest. This means that the conclusion holds for fixed small sample sizes, although the constant involved may be quite large. Intuitively, one would expect that F_m is a better approximation to F for larger values of m, and indeed this is so in the characterization of Lemma 3.7. However, both m-1 and $X_{(m)} - X_{(1)}$ may increase with m, apparently making the factor in Lemma 3.8 larger. This is not really surprising, since the bound is a worst case one, and on the average it should really be true that the ECDF becomes a better approximation as m grows.

The primary application of the ECDF is in the familiar "divide-and-conquer" framework. The usual divide-and-conquer strategy results in a recurrence of the form

$$T(n) = 2\,T(n/2) + D(n) + M(n)$$

where D(n) is the time required to "divide" the original problem into two subproblems (each of size n/2) and M(n) is the time required to "marry" the two subproblem solutions to get the solution to the original problem. Typically, $D(n) = O(n)$ and $M(n) = O(n)$, so that $T(n) = O(n \log n)$. If D(n) and M(n), or their expected values, are $O(n^h)$ for some $h < 1$, then the divide-and-conquer scheme yields a linear-time algorithm (see Bentley and Shamos [1978]).

Here we take a slightly different approach to produce linear expected time algorithms using divide-and-conquer. Briefly, the idea is to try to simultaneously divide the original problem into O(n) subproblems, each of approximately the same size, and then to marry the subproblem solutions together. The resulting recurrence is approximately of the form

$$T(n) = (n/C)\,T(C) + D(n) + M(n)$$

which has the solution $T(n) = O(n)$ whenever $D(n)$ and $M(n)$ are each $O(n)$.

For the purposes of the following procedure and lemma, which introduce the basic method and which will be illustrated and expanded in Section 3.4, we assume as usual that the problem to be solved is specified by a set $S = \{X_i\}$ of n real numbers which are independent observations on a random variable having the (possibly unknown) distribution function F(x). If F is known, we assume that it can be computed in constant time. If it is not known, then we substitute for it the ECDF F_m, where m is a constant independent of n. It is necessary to redefine $X_{(1)}$ and $X_{(m)}$ to be the minimum and maximum of all n inputs, respectively, rather than the minimum and maximum of the sample of size m, a change which does not affect the validity of any of the lemmas we have proved. It is clear that $F_m(x)$ is computable in constant time.

In the procedure "solve" below, H(x) is the transformation described in Lemma 3.6; if F is unknown, this amounts to using just $F_m(x)$. C and ν are constants.

```
procedure solve(S);
  set S;
  if |S| ≤ ν then solve S by brute force else
    begin
      integer n = |S|;
      set array R[0:n/C] = φ;
      for each element x ∈ S do
        begin
          integer j;
          j := ⌊H(x)*n/C⌋;
          R[j] := R[j] ∪ x
        end;
      for j := 0 to n/C do solve R[j] by brute force;
      marry together solutions to R[0], ..., R[n/C]
    end;
```

Let $T(n)$ be the expected time to solve a problem of size n by this procedure[3]; let p_j be the probability that $\lfloor H(x)*n/C \rfloor = j$; and let $k_j = |R[j]|$. Suppose that the brute force solution of the problem can be accomplished in polynomial time $T_0(n)$, so that $T_0(n) \le \sum_i a_i n^i = \sum_i b_i \binom{n}{i}$, where $a_i = b_i = 0$ for $i > K$, and that the marriage step takes linear expected time. Then the total solution time is given by

$$T(n) \le \sum_j T_0(k_j) + An + B$$

for some constants A and B. Averaging over the distribution p_j of inputs into subproblems, we have

$$T(n) \le \sum_j \sum_k T_0(k)\, P\{k_j = k\} + An + B$$

---

3 . This average is over the distribution F, and makes no assumptions about the distribution of permutations of the inputs; they could be presented to the algorithm in sorted or shuffled order without affecting the running time. In fact, all that is really required is that the X_i are identically distributed.

$$\leq \sum_j \sum_k \sum_i b_i \binom{k}{i} \binom{n}{k} p_j^k (1-p_j)^{n-k} + An + B .$$

We are now ready to describe sufficient conditions on F for T(n) to be linear.

LEMMA 3.9 -

If F is known, or if F satisfies a Lipschitz condition and is the distribution function of a bounded random variable, then $T(n) = O(n)$.

Proof - For some constants A and B,

$$T(n) \leq \sum_j \sum_k \sum_i b_i \binom{k}{i} \binom{n}{k} p_j^k (1-p_j)^{n-k} + An + B$$

$$= \sum_i b_i \binom{n}{i} \sum_j p_j^i + An + B$$

Now,

$$p_j = P\{ \lfloor H(X)n/C \rfloor = j \}$$

$$= P\{ (j-1)C/n \leq H(X) < jC/n \}$$

If F is known, then by Lemma 3.6, $p_j = C/n$ for $0 \leq j < n/C$. On the other hand, if F is unknown, but satisfies a Lipschitz condition and is the distribution function of a bounded random variable, then by Lemma 3.8 there is a constant $\gamma \geq 1$ such that

$$p_j = F(F_n^{-1}(jC/n)) - F(F_n^{-1}((j-1)C/n))$$

$$\leq \gamma C/n$$

for $0 \leq j \leq n/C$. An example of this situation is illustrated in Figure 3.3. In either case,

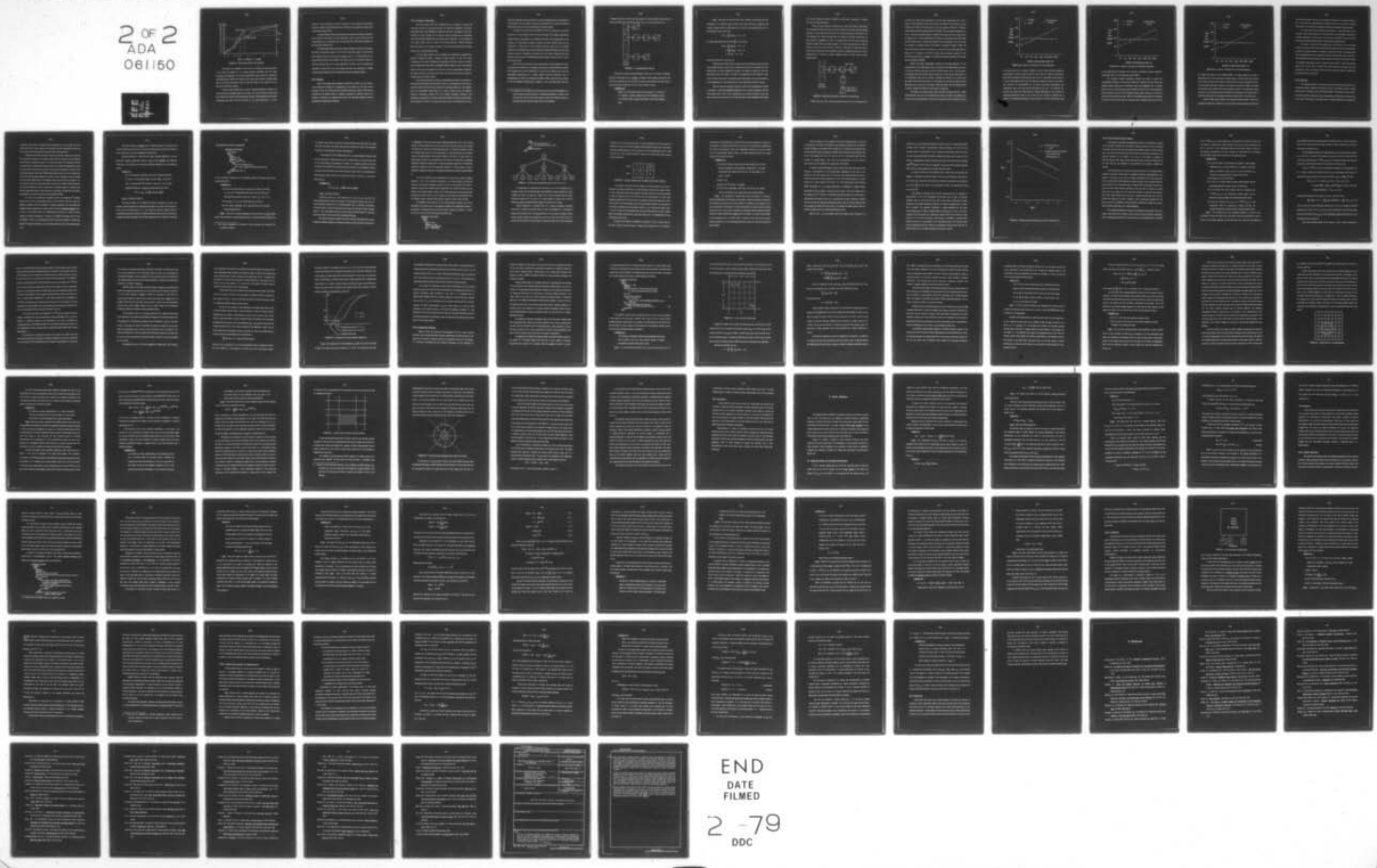
$$T(n) \leq \sum_i b_i \binom{n}{i} (\gamma C/n)^{i-1} + An + B$$

$$\leq \left(\sum_i b_i (\gamma C)^{i-1} + A \right) n + B$$

Since $\sum_i b_i (\gamma C)^{i-1}$ is bounded and independent of n, the lemma is proved. □

AD-A061 150 CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER --ETC F/G 12/1
STATISTICAL METHODS IN ALGORITHM DESIGN AND ANALYSIS. (U)
AUG 78 B W WEIDE N00014-76-C-0370
UNCLASSIFIED CMU-CS-78-142 NL

2 OF 2
ADA
061150



END
DATE
FILMED
2-79
DDC

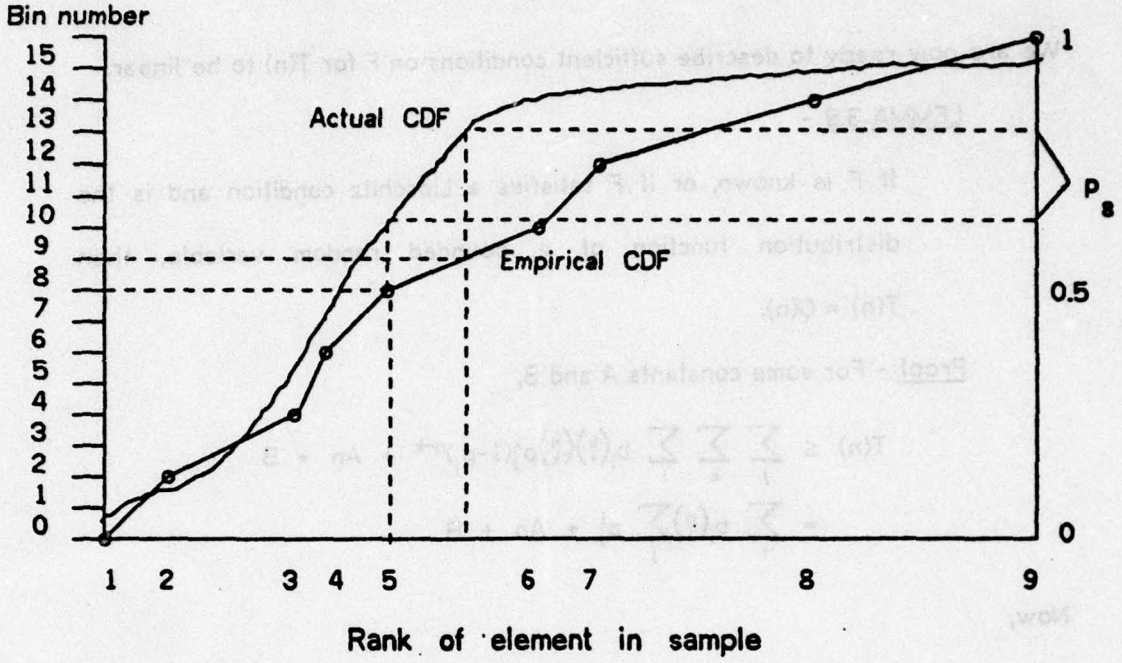


FIGURE 3.3 - Relationship between F and the ECDF.

The result is true regardless of the value of m. Incredibly, this means that even if we treat the elements of S as being uniformly distributed over the range (X_{min}, X_{max}) the algorithm runs in linear expected time. Larger values of m, resulting in presumably better ECDF's and therefore more nearly equal division of elements, also lead to increased computation times for H(x). The effect of this trade-off will be different for different problems.

The first major difficulty here is that for unknown distribution functions, one can test neither assumption which we have made. This is still a big step forward from the restriction to uniformly distributed inputs, however. That assumption can be tested statistically, but usually with the result that it is not valid! Furthermore, it is often

possible to take advantage of special circumstances of the problem to expand the domain of validity of the conclusion that T(n) = O(n). One such problem is sorting, which is discussed in Section 3.4.1.

A second difficulty arises when applying the procedure to problems defined by sets of points in the plane or in higher dimensions. Then it is not so obvious how to compute H(x), nor is it necessarily easy to show that M(n) = O(n). These problems will be discussed in Section 3.4.4.

It is interesting to note that "solve" could be applied recursively to the smaller sets R[j]. The expected number of such recursive calls which would not use the brute force method is equal to n/C times the probability that $k_j > \nu$. The distribution of k_j is approximately Poisson with parameter C for large n, so even for moderate values of C and the ratio ν/C, $P\{ k_j > \nu \}$ is very small. With C = 5 and $\nu = 15$, for example, this probability is about 10^{-4}, so on the average only about one subproblem in 10,000 would be large enough to warrant solution by the more sophisticated method.

3.4. Examples

Each of the design tools proposed in Section 3.3 is based on a few lemmas which indicate the domain of its applicability. In this section we will give several examples of the use of these tools and associated methods of analysis which help to demonstrate the potential utility of statistical techniques for algorithm design. The problems considered are organized into sorting and searching, selection, discrete optimization, and geometrical problems.

3.4.1. Sorting and Searching

We have already seen how randomization can be applied to sorting and searching problems to give good expected time algorithms regardless of the input distribution (see Yuval [1975b] for Quicksort and Carter and Wegman [1977] for hashing). For these problems, as for most, the model of computation can dramatically affect the problem complexity and the running times of algorithms. In the comparison tree model usually used for sorting and searching, sorting n elements requires $\theta(n \log n)$ steps on the average as well as in the worst case. Searching an ordered table of size n requires $\theta(\log n)$ steps.

It is not clear, however, how to implement the randomization suggested for Quicksort without the ability to generate random numbers, as well as to make comparisons. In the case of hashing, the computation of the floor function in constant time is assumed. In fact, Carter and Wegman [1977] employ an even more specific model in which keys are elements of a finite set A which are represented as $\lceil \log |A| \rceil$-bit strings. Sorting can be done in linear time in the worst case under such a model by a direct address calculation method.

It is obviously desirable to choose a computational model which is as general as possible while maintaining the relevance of results to real applications. To this end, we assume as before that the inputs to a sorting or searching problem are n real numbers which are independent observations on a random variable having the (possibly unknown) distribution function F(x). The standard arithmetic operations and comparisons, computation of the floor function, generation of a uniform random deviate, and computation of F(x) (if F is known) each take unit time. No assumptions are made

about how quantities are represented on a machine, although results can sometimes be strengthened if we are willing to make such assumptions.[4] This is the same model of computation which we have used throughout the present chapter.

It should be noted that the availability of the floor function as a primitive operation is a very powerful tool. In the usual comparison tree model, computing $\lfloor x \rfloor$ requires $\theta(\log x)$ comparisons. As a consequence of the radix representation of real numbers on actual digital computers, however, the floor function really can be computed in constant time on such machines, and there seems to be no practical reason for not allowing its use. The fact that lower bounds are difficult to prove when the floor function is computable in constant time is only mildly irritating because the expected time of many of the algorithms which we propose is linear, and this is also a trivial lower bound.

We first show how sorting can be accomplished in linear expected time under rather general conditions on F by applying the procedure "solve" presented in Section 3.3.4. Suppose that we have a linked list of n records, to be sorted by keys which are independent observations on a random variable having the distribution F(x). In implementation of "solve" for sorting, each set R[j] is simply a linked list, with a pointer to the first element of R[j] stored in position j of an array (see Figure 3.4). The

---

4. For example, we may assume that the items being sorted are simply elements of a totally ordered set, and that there is a mapping (computable in constant time) from that set into the reals which preserves order. This evidently applies to such data types as character strings on most real computers.

marriage step simply involves marching through the sorted list R[0], chaining the last element of R[0] to the first element of R[1], and so on, which takes linear time.

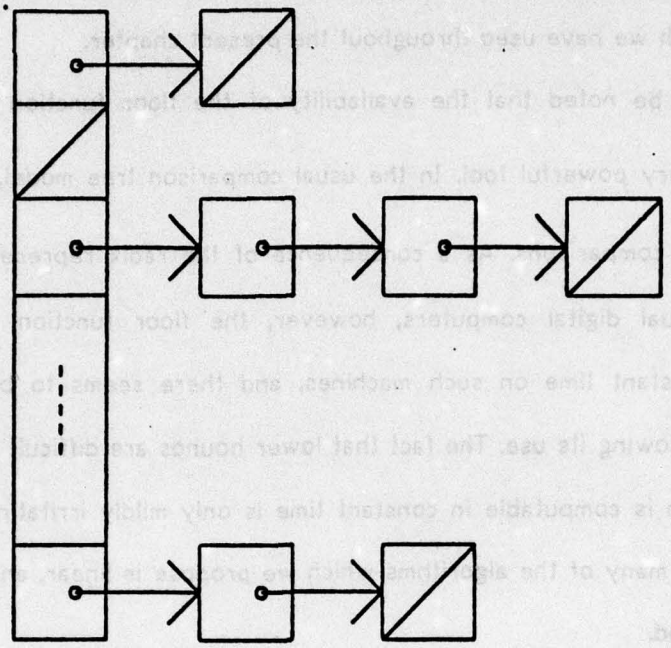


FIGURE 3.4 - List implementation of Binsort.

Because this method essentially divides the input set S into "bins" or "buckets" during its initial phase, we call it Binsort. Although an $O(n^2)$ method will suffice as the brute force algorithm for the lists R[j], it is good insurance against the worst case to use an O(n log n) method here. For this version of Binsort we have

THEOREM 3.10 -

Binsort runs in O(n log n) time in the worst case. If F is known, or if F satisfies a Lipschitz condition and is the distribution function of a bounded random variable, then Binsort runs in O(n) expected time.

Proof - We show only that the worst case of Binsort is O(n log n), since the remainder of the theorem follows directly from Lemma 3.9. Let $k_j = |R[j]|$ be the number of items which are distributed to bin j. If T(n) is the computation time for this distribution of items to bins, then

$$T(n) \leq \sum_j Dk_j \log k_j + An + B$$

for some constants A, B, and D. Since $\sum_j k_j = n$, we have

$$\begin{aligned} T(n) &\leq \sum_j Dk_j \log k_j + An + B \\ &\leq D\sum_j k_j \log n + An + B \\ &= D\, n \log n + An + B \end{aligned}$$

which proves that T(n) = O(n log n). □

A special case of Binsort which assumes uniformly distributed inputs and which uses straight insertion as the brute force method for small lists is presented by Knuth [1973] under the name "multiple list insertion". He includes an analysis for uniformly distributed inputs, and credits R. M. Karp with suggesting that the algorithm would work in linear time for other "sufficiently smooth" distributions. The use of the ECDF F_n is not suggested, nor is the extension to any known distribution, however irregular.

The worst case of multiple list insertion is $O(n^2)$, which led Dobosiewicz [1978] to propose a much more complicated algorithm and an equally complicated analysis which shows that it runs in linear expected time for uniformly distributed inputs and has an O(n log n) worst case. It is now clear that only a minor modification of multiple

list insertion (using an O(n log n) method for small files) is necessary to achieve O(n log n) worst case behavior.

There are many possible refinements that would make Binsort theoretically more desirable, but that can actually increase sorting time in practice. One such change accounts for equal elements, and allows us to extend the domain of distributions for which Binsort runs in linear expected time. This change is illustrated in Figure 3.5. As each element is assigned to a bin, it is inserted into its proper place among the elements already there by straight insertion. If it is found to be equal to an element already in the bin, the new element is added to a "side chain" of elements, each of which has the same value. This modification results in an algorithm which runs in linear expected time if F satisfies a Lipschitz condition except for a finite number of jump discontinuities.

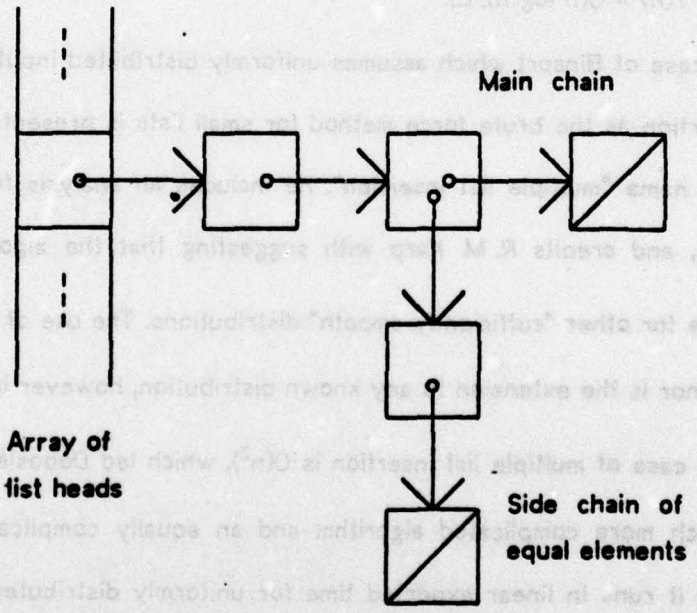


FIGURE 3.5 - Modification of Binsort to account for equal elements.

Binsort has many other desirable properties which will not be discussed here

in detail, but which will be developed in a future paper. Among these are: (1) If a stable sorting algorithm is used for small lists, then Binsort is stable (that is, equal elements are output in the same order in which they appear in the input). (2) Binsort seems ideal for external sorting (see Knuth [1973]). The most obvious approach is to use it in the internal sorting phase of a standard sort/merge, but there is good reason to believe that a direct implementation of Binsort for extremely large files would perform better. (3) On a multiprocessor, Binsort can be implemented in such a way that the speedup is essentially equal to the number of processors available. It does not seem to be easy to achieve this (except as $n \to \infty$) for the usual sorting algorithms. In Quicksort, for example, it is difficult to make several processes cooperate in the partitioning step, while in mergesort it is difficult to make them cooperate in the merge step (see Robinson [1978]).

Binsort has been implemented in Bliss-10 and timings gathered for this implementation on the DECSystem-10 (KL-10 processor) at Carnegie-Mellon University. The results of these experiments are shown in Figures 3.6(a, b, and c). The constant C is 5, meaning that the average number of elements in a bin is 5; the sample size m is 13 intermediate items plus the extrema. Although the elements to be sorted were generated from known distributions, this information was not given to Binsort, which therefore had to rely only on the ECDF. Small files (fewer than 20 elements) were sorted by straight list insertion for both Binsort and Quicksort.

All timings are average times, shown with 95% confidence intervals. Figure 3.6(b) indicates that even for a distribution which does not satisfy the conditions for which we can prove that Binsort runs in linear expected time, the non-linearity (if any)

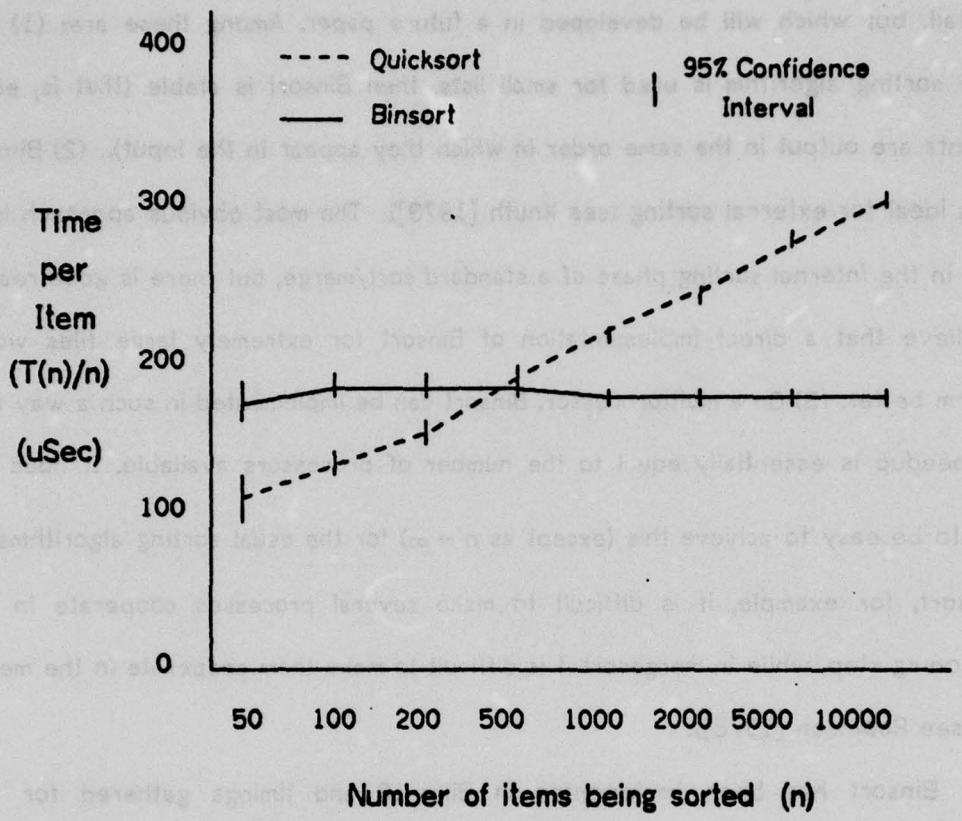


FIGURE 3.6 (a) - Binsort vs. Quicksort for uniform distribution.

is not serious even for n as large as 10,000. It is also of interest that the constant of proportionality is almost exactly the same for these three very different distributions, about 180 microseconds of CPU time per item in the list. With smaller sample sizes (5, 1, and 0 intermediate points, plus the extrema) the algorithm performs even better. The constants of proportionality for all three experimental distributions are, respectively, about 163, 133, and 106 microseconds per item. This indicates that perhaps the sample size itself should be adjusted depending on how smooth the underlying distribution seems to be, so that the time required to compute H(x) is smaller if the underlying distribution is nearly uniform. For the three distributions used

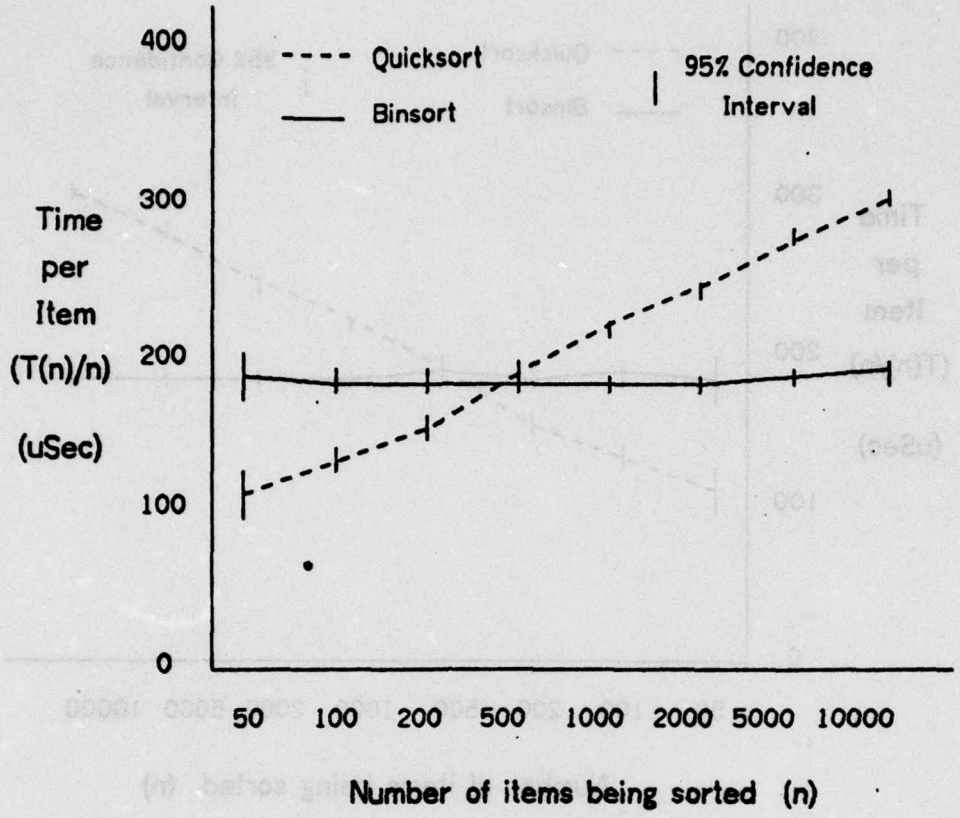


FIGURE 3.6 (b) - Binsort vs. Quicksort for exponential distribution.

in our experiments, the deviation from uniformity is apparently not great enough to warrant the use of a very sophisticated approximation.

For extremely irregular populations, however, the ability to adjust the approximating function is an important practical tool. Consider the case where the elements being sorted have a distribution which with probability 1/2 is uniform between 0 and 1, and otherwise is uniform between 100 and 101. For that distribution, Binsort performs best with 29 intermediate sample points, achieving a constant of about 206 microseconds per item. With no intermediate sample points, the expected run time is more than four times as long, since only a relatively few bins ever have

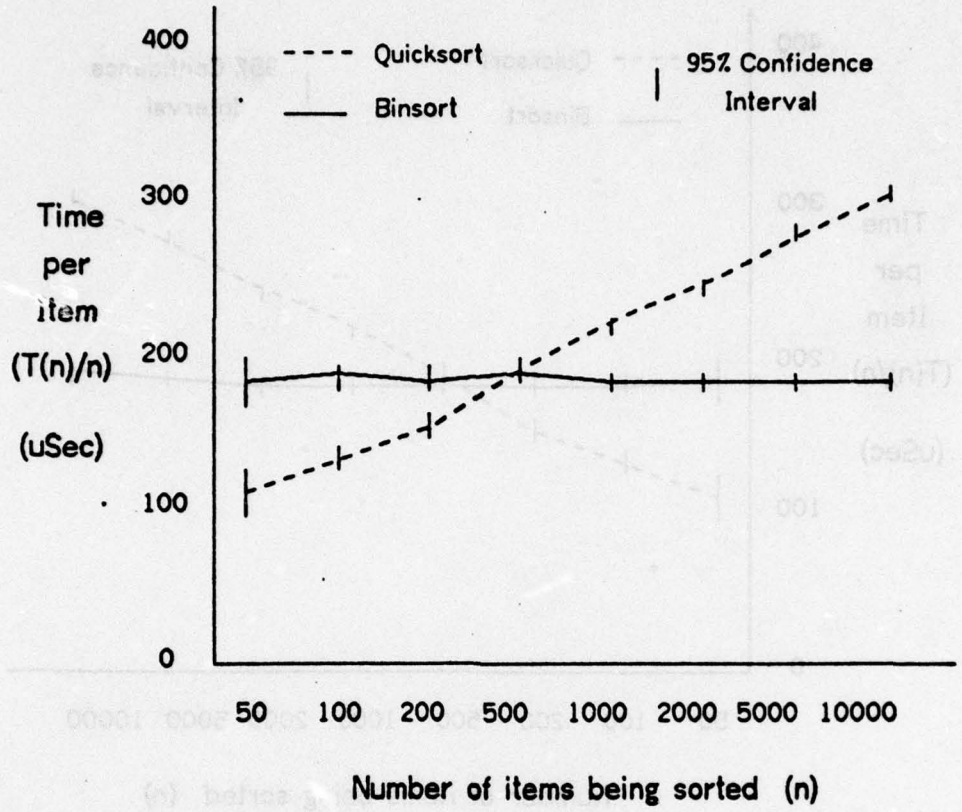


FIGURE 3.6 (c) - Binsort vs. Quicksort for a tri-modal distribution.

any items. The effect of the improved ECDF is so clearly apparent on such a distribution that it seems desirable to choose initially a relatively large (but constant size) sample, compute some statistic to determine how much the population differs from uniformity, and keep a fraction of that sample to calculate the ECDF. The larger the deviation from uniformity, the more elements are kept. This modification seems essential in any production version of Binsort, but would require more experiments to determine how large the initial sample should be and what part of it should be kept.

Binsort handily beats Quicksort even though the former makes an initial pass through the entire list to sample for F_m. For the data in the experiments, this would not

have been necessary since the first m elements constitute just as good a sample as any m elements. The implementation of Quicksort makes use of this fact, however, choosing the first element for partitioning and thereby avoiding the sampling passes.

It is clear from these experiments that Binsort is a viable alternative to the commonly used sorting algorithms (of which Quicksort is the most highly regarded for practical applications). Furthermore, it is fairly easy to implement and has the other potential advantages mentioned above which should be explored further.

The idea of using the ECDF applies even more directly to the searching problem. If we have a set of items to organize for searching, we can use an addressing function to map each element to a position in a table. This is similar to hashing, except that the addressing function is designed to organize the data rather than randomize it. One function which works under very general conditions (those of Lemma 3.9) is simply F_m. Expected preprocessing time to organize the hash table is linear and expected lookup time for any item is constant. This result should be contrasted with that obtained by Carter and Wegman [1977] under the different model of computation described earlier in this section.

3.4.2. Selection

The selection problem, in its abstract form, has received much attention from theoretical computer scientists. The abstract version of the problem is: Given a totally ordered set of n elements, find the k^{th} smallest element. The problem requires and can be solved using $\theta(n)$ comparisons on the average (Hoare [1962]) and $\theta(n)$ comparisons in the worst case (Blum, et al. [1973]). Floyd and Rivest [1975] have presented an

algorithm which uses $n + \min(k,n-k) + o(n)$ comparisons on the average, and have shown that this is nearly optimal. For the median, the most interesting case and the one which we will concentrate on, this gives $3n/2 + o(n)$ comparisons.

Despite the fact that so much is known about the time complexity of selection, there are other aspects of the problem which have not received as much attention. One of these is the space required. It is known that even if the elements of the set are generated or transferred to primary memory of a computer sequentially, we must still have direct access to $n/2$ elements at some stage of the algorithm in order to find the median element (see Knuth [1973], section 5.3.3, exercise 15). In this section we show how the techniques suggested in Section 3.3 make it possible to take advantage of the "on-line" appearance of the elements, saving some direct access storage, if we are willing to settle for a good approximate answer to the problem. Unfortunately, for the on-line algorithms we must assume that the elements appear in random order, since randomization does not seem possible in this setting. The algorithms and results presented here have been previously reported (Weide [1978]).

It is clear from considering the possible reasons for finding the k^{th} smallest element of a set that an approximate answer is often sufficient. There are really two separate problems, the first essentially a statistical application and the second the computer science abstraction of it. We call the first problem quantile estimation: Given a set of n real numbers which are independent observations on a random variable having an unknown distribution F, estimate the $(100\alpha)^{th}$ percentage point of the distribution. There is no requirement that the estimate be obtained by finding the $(\lfloor \alpha n \rfloor + 1)^{th}$ smallest observation, or even that the estimate be one of the observations at all.

The second problem is selection: Given a totally ordered set of n elements, find the k^{th} smallest element. Here we do require that the answer be one of the elements of the original set, for reasons suggested in Section 3.3.2.

Quantile estimation is usually done by using a selection algorithm to find the $(\lfloor \alpha n \rfloor + 1)^{th}$ smallest observation, which is used as the estimate. The statistical properties of this estimate are well known; of particular interest to us is the following lemma.

LEMMA 3.11 -

Let F be absolutely continuous and strictly increasing whenever $0 < F(x) < 1$; let the density $f(\xi_\alpha) > 0$, where $F(\xi_\alpha) = \alpha$; and let F have a finite absolute δ^{th} moment for some $\delta > 0$. If X_n is the $(\lfloor \alpha n \rfloor + 1)^{th}$ smallest of n independent observations from F, then

$$n^{1/2} (X_n - \xi_\alpha) \rightarrow_d \mathfrak{N}\left(0 , \alpha(1-\alpha) / f(\xi_\alpha)^2\right)$$

Proof - See Walker [1968]. □

This lemma enables us to determine how good an estimate X_n is to ξ_α. The difficulty with the algorithm for producing the estimate is the need for direct access to up to n/2 of the observations. If n is very large, the resources required might be unacceptably expensive. We seek an alternative estimate which uses less direct access storage and takes advantage of the on-line appearance of the elements of the set;

subsampling (Section 3.3.3) is applicable[5].

```
procedure estimate(S,α);
    set S; real α;
    begin
        set Q;
        real total = 0;
        integer j, n = |S|;
        for j := 1 to n/r(n) do
            begin
                Q := next r(n) elements of S;
                total := total + ((⌊αn⌋+1)th smallest element of Q)
            end;
        return( total * r(n) / n )
    end;
```

For the procedure "estimate", we can immediately deduce the following theorem from Lemmas 3.5 and 3.11.

THEOREM 3.12 -

Let Z_n be the result obtained by applying the procedure "estimate" to a set S of n independent observations from the distribution F satisfying the conditions of Lemma 3.11. Then if $r(n) \to \infty$ as $n \to \infty$,

(a) $Z_n \to_{pr} \xi_\alpha$; $Z_n - X_n \to_{pr} 0$; and $(Z_n - X_n)/X_n \to_{pr} 0$ if $\xi_\alpha \neq 0$.

(b) The on-line computation of Z_n uses $\theta(n)$ time and r(n) direct access storage locations.

Proof - Part (a) is a direct consequence of the fact that $X_n \to_{pr} \xi_\alpha$, which follows from Lemma 3.11, and Lemma 3.5 with h(n) = 1. Since the selection problem can

---

5 . Mike Shamos suggested the problem of on-line selection and proposed the procedure "estimate".

be solved in linear time, the first part of (b) also follows from Lemma 3.5. It is easily seen that only about r(n) direct access storage locations are used in the procedure "estimate" if the elements of S appear sequentially. □

This theorem not only demonstrates that Z_n is a good estimate of ξ_α, but also gives an algorithm for approximating X_n which succeeds weakly. It is easy to show that for this interpretation (considering the problem as a selection problem where the inputs are real numbers) the algorithm succeeds strongly in the independent model if r(n) grows faster than log n. But while the theorem tells us that Z_n is asymptotically a good estimate of ξ_α, it falls short of telling us how good it is. We can, in fact, prove more about Z_n.

THEOREM 3.13 -

$$n^{1/2}\,(\,Z_n - \xi_\alpha\,) \rightarrow_d \mathfrak{N}\big(0\,,\ \alpha(1-\alpha)\,/\,f(\xi_\alpha)^2\big)$$

Proof - See Weide [1978]. □

Combining Lemma 3.11 with Theorems 3.12 and 3.13, we see that the subsampling algorithm produces an estimate of the population quantile which has the same asymptotic distribution as the sample quantile, finds the estimate in essentially the same amount of time, but operates on-line using only r(n) direct access storage, where r is any unbounded function of n and can grow arbitrarily slowly. It is obvious that this problem is well-suited to the subsampling technique.

The quantile estimation problem shows that the lower bound on direct access storage for selection applies to the abstract problem, but not to its natural counterpart

in application. If we must really solve a selection problem, for any of the reasons offered in Section 3.3.2, then we must be prepared to pay the price for that storage requirement, or settle for an approximation algorithm. An interesting situation in which the problem might arise is in the organization of a large data base for fast queries (see, for example, Bentley and Friedman [1978]). In this case, as in many others, an approximate answer (i.e., an element with rank close to that desired) is acceptable, especially in light of the alternative. An easy approximation algorithm for the selection problem is simply to find the appropriate element of a small sample of, say, m elements. Lemmas 3.3 and 3.4 can be applied to determine how good the approximation is.

An on-line algorithm can be implemented by first generating m distinct integers between 1 and n, then sampling the elements in just those positions of the input stream. It is somewhat unsatisfying, however, to use time which is linear in n, which is necessary because the elements appear sequentially, but to ignore perhaps the vast majority of the data available. In light of this, it seems desirable to develop some mixed strategy which makes some use of each element in determining the approximation, but only needs to keep a relatively small number of elements in direct access storage.

An algorithm which does this for the median selection problem, and can be modified for the general case at the expense of slightly more complicated analysis, is given below. The general idea is to find the "median of medians of medians ..." in order to conserve space without increasing the computation time.

```
procedure median_est(S);
   set S;
   begin
      set Q, T = φ;
      integer j, n = |S|;
      if n = 1 then return(S);
      for j := 1 to n/m do
```

```
        begin
            Q := next m elements of S;
            T := T ∪ (exact median element of Q)
        end;
    return(median_est(T))
end;
```

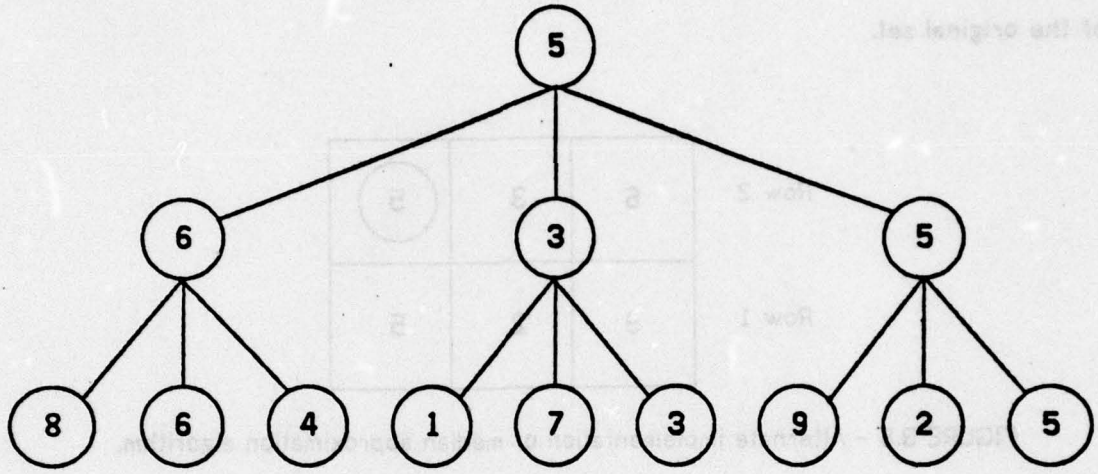


FIGURE 3.7 - An example of median approximation with n = 9, m = 3.

The operation of this algorithm for the case n = 9, m = 3 is illustrated for an example in Figure 3.7. This special case of the algorithm has been proposed by Tukey [1978], who calls the result the "ninther". He is more concerned with the good statistical properties of the ninther as a robust estimate of location than with the algorithmic aspects, although he does suggest its potential on-line nature.

While "median_est" is presented here as a recursive procedure operating "bottom-up" to make its operation clear, it is easy to implement it in a "top-down" fashion which works on-line. This approach uses an array of size m by $\log_m n$ in direct access storage. Elements are read into the first row of the array until it is full, at which point the median of those elements is found and placed in the next unoccupied

position of row 2, if there is one. Row 1 is cleared (logically) for the next group of m elements. If row two is full, the median of those elements is determined and placed in the next available position of row 3, causing row 2 to be cleared, and so forth. When the last row is full, its median element is reported as the approximate median element of the original set.

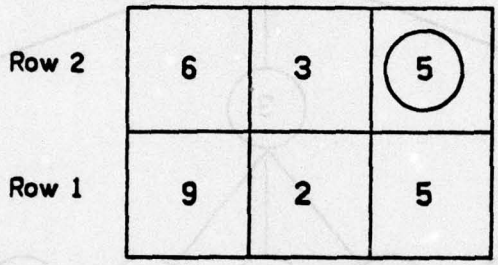


FIGURE 3.8 - Alternate implementation of median approximation algorithm.

The state of the array after the algorithm has been applied to the data in Figure 3.7 is shown in Figure 3.8. The circled element ranked 5 is the median of those in row 1 (the last group of three elements in the input stream); the previous elements in row 2, ranked 6 and 3, are from the first and second groups of three input elements, respectively. The final answer is the element ranked 5 (the median of row 2), which luckily in this case happens to be the exact median element of the original set. It is easy to determine that the element ranked 4, 5, or 6 must be the answer when $n = 9$, $m = 3$, and that the probability that the exact median element is found is almost 60%. The example illustrated may have been lucky, but it is representative of the possibilities for this case.

Analysis of the algorithm presented is not difficult. The time is easily seen to be linear, the direct access storage is $m \log_m n$, and the expected rank of the element

produced as the approximation is equal to the true median. Bounding the variance of the estimate is somewhat more difficult, since the ranks of the elements produced for the second and subsequent rows are not independent. However, we can get a result which is asymptotically valid for large n, and which experiments show is an upper bound on the variance for small n.

THEOREM 3.14 -

Let Z_n be the relative rank (i.e., the true rank divided by n+1) of the element produced by the procedure "median_est". If all input permutations are equally likely, then for any odd integer $m > 1$, with $m = o(n^{1/2})$,

(a) $E(Z_n) = 1/2$.

(b) $D(Z_n) = O(n^{-1+h})$, where $h = \log_m(\pi/2)$.

(c) The on-line computation which finds the element with relative rank Z_n uses $\theta(n)$ time and $m \log_m n$ direct access storage locations.

Proof - The assumption of equally likely input permutations is necessary because randomization of the inputs is not possible. It affects only parts (a) and (b) of the theorem, since the time and space used by the algorithm do not depend on the input permutation. Furthermore, for certain permutations of special interest, including inputs which are sorted in either increasing or decreasing order, the algorithm actually produces a better estimate than suggested by the theorem. The same remark applies to the assumption that $m = o(n^{1/2})$, which is included so that Lemma 3.4 will apply without modification. It is to be expected that larger values of m will only improve the approximation.

Part (a) is clearly true by symmetry since we have specified that m is odd and are looking for the median. The difficulty with adapting the algorithm for the general selection problem is encountered here. Suppose we are looking for the element with relative rank 1/3; then we choose the element ranked (m+1)/3 in each sample of size m, but take medians the rest of the way up the tree. The desired result that the estimate is unbiased ($E(Z_n) = 1/3$) is only true asymptotically as $m \to \infty$. No such restrictions on m are needed for the special case of the median.

In order to bound the variance of Z_n, we note that for large n the ranks of the elements in each group of m are approximately independent of the ranks of the elements in the other groups. There is, of course, some interaction, because there is exactly one element with each possible rank, and if that element appears in a certain group of m it cannot appear in the others. As a first approximation which becomes better and better as $n \to \infty$, though, each group of m elements is a sample without replacement from the original n. Letting $X_{1,i}$ be the relative rank (in the original set) of the median of the i^{th} sample of m elements, and applying Lemma 3.4, we find that an appropriately normalized form of $X_{1,i}$ is asymptotically normally distributed. Another way to overcome the dependence among the groups, which is even less rigorous than this argument but leads to the same result, is to consider the relative ranks to be i.i.d. random variables which are uniformly distributed over (0,1).

Now if we let $X_{2,i}$ be the relative rank of the median of the i^{th} group of m

medians $X_{1,j}$ we are essentially finding the median of a set of m independent random variables each of which is approximately normally distributed with mean 1/2 and variance 1/(4m). The distribution of $X_{2,j}$ is therefore asymptotically normal with mean 1/2 and variance $2\pi/(16m^2)$ (see Walker [1968]). Continuing in this manner, we find that Z_n is asymptotically normally distributed with mean 1/2 (see part (a)) and variance $(\pi/2)^{\log_m n}/(2\pi n) = O(n^{-1+h})$ where $h = \log_m(\pi/2)$. Figure 3.9 provides evidence that this bound on the variance applies even for small values of n and m.

In order to prove that the algorithm runs in linear time, we note that the median of m elements is found n/m times in the first row of the array, n/m^2 times in the second row, and so forth. The total time is therefore at most $\theta(m)$ for each median of m, times $n\sum_i m^{-i}$, for a total of at most $\theta(nm/(m-1)) = O(n)$. The space bound in part (c) is obvious. □

The methods of Sections 3.3.2 and 3.3.3, involving the use of sampling for approximation in rank and subsampling, are obviously easy to apply to selection problems. Here we have seen that they lead to new results concerning the space necessary to find good approximate solutions for selection problems, both in theory and in practice. The algorithm "median_est" is being considered by the image understanding group at CMU for implementation in LSI chips for n = 25, m = 5. This is especially attractive because one "approximate median of 25" circuit is made of 6 small "median of 5" circuits, and because a high degree of parallelism is possible. Mike Shamos has given an implementation of an "approximate median of 25" network which uses 42 comparators and has a delay of 10 comparators, and has shown that with no fanout a delay of 5 for finding the median of 5 elements is optimal.

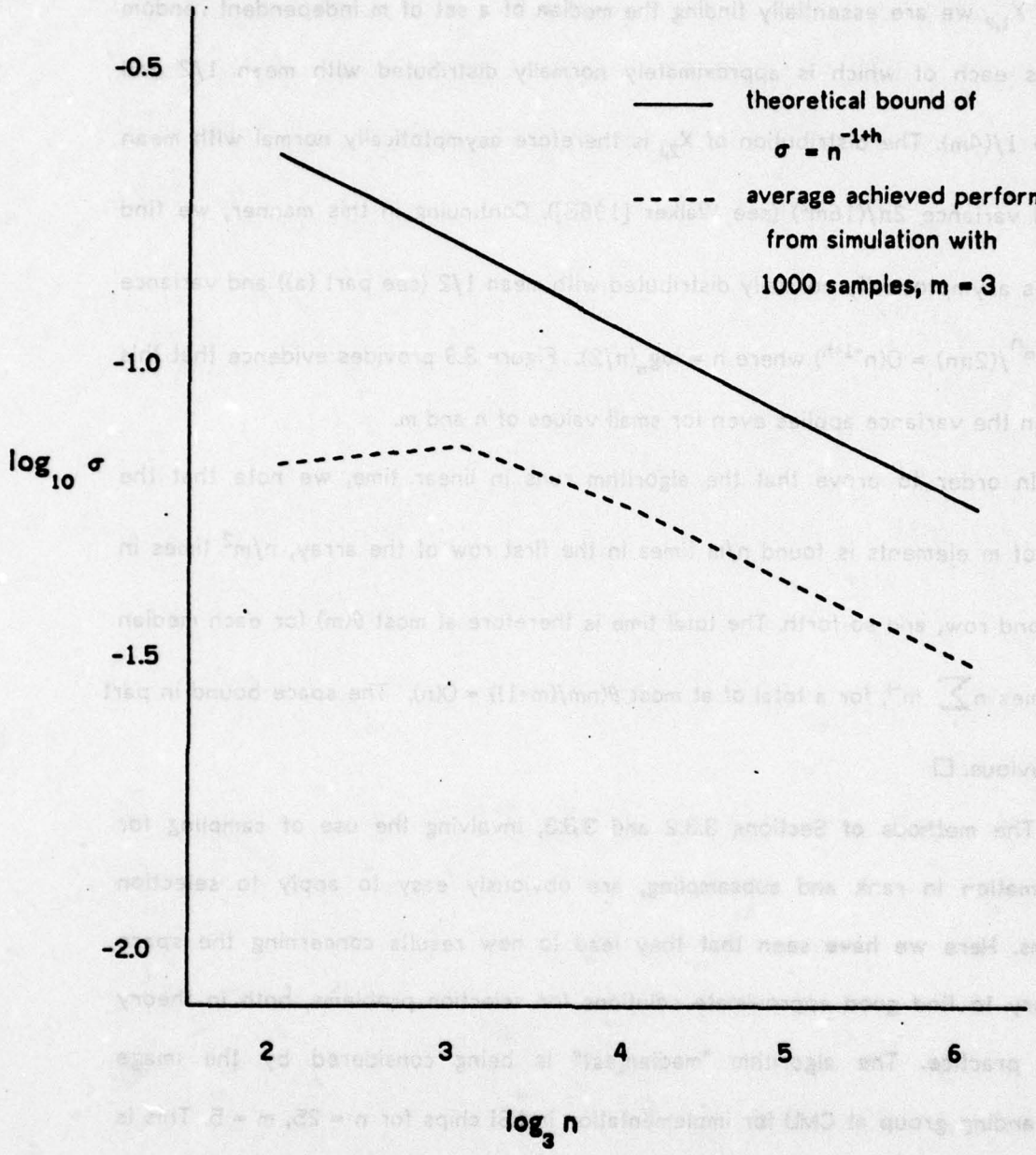


FIGURE 3.9 - Experimental substantiation of the bound of Theorem 3.14.

3.4.3. Discrete Optimization Problems

This section is devoted to the application of some of the techniques of Section 3.3 to discrete minimization and maximization problems. Until now, all the applications have been obvious, since we were dealing with problems defined on totally ordered sets. In Sections 3.4.3 and 3.4.4, the fact that there are ordered sets somewhere in the problem definition or its solution is not clear at first glance. In addition to demonstrating the wide range of utility of statistical design and analysis techniques, we will also see how the analogues of non-parametric and parametric methods can interact.

One problem which is currently of universal interest is the traveling salesman problem. In Chapter 2, we saw how it is possible to construct a respectable tour very quickly if the points to be visited are chosen independently from some distribution over the plane. Karp's [1977] algorithm succeeds strongly in the incremental model for any distribution over a bounded region. As Golden [1977] has pointed out, however, there are many other heuristic approaches which also seem to produce very good tours, also very quickly, but which cannot be analyzed to prove that they perform well. One would therefore like some way to predict approximately how long the optimal tour should be in order to compare it with a heuristically produced tour. We first show how it is possible to accurately predict the optimal tour length very quickly, without actually producing a tour, using subsampling.

The immediate application of Lemma 3.5 is possible for this problem. If X_n is the length of the optimal tour through n points chosen from some (unknown) distribution over a Lebesgue-measurable set in k-space, then we know (Beardwood, et al. [1959])

that $n^{-(k-1)/k}X_n \rightarrow_{pr} \mu$, where μ depends on the distribution of the points. Actually, the convergence is almost sure in the incremental model, and we have conjectured that this is also true in the independent model but have been unable to prove it. Convergence in probability is all that is required for Lemma 3.5, and in the general case the ETSP satisfies all the conditions for subsampling to apply.

THEOREM 3.15 -

Let X_n be the length of the optimal tour through n points chosen independently from some distribution over a Lebesgue-measurable region of Euclidean k-space, and let Z_n be the estimate of X_n produced by subsampling with n/r samples each of size r.

(a) This algorithm runs in $O(nr2^r)$ time.

(b) If r(n) is any function which increases without bound, then the subsampling algorithm succeeds weakly in estimating X_n.

(c) If r(n) satisfies the further condition that $r(n) = o(n/\log n)$, then $n^{-(k-1)/k}Z_n \rightarrow_{as} \mu$ in the independent problem model, and the algorithm succeeds strongly in the incremental problem model.

(d) Under the conditions of part (c), if $n^{-(k-1)/k}X_n \rightarrow_{as} \mu$ in the independent model (as conjectured in Chapter 2), then the subsampling algorithm succeeds strongly in the independent model.

Proof - The running time of the subsampling algorithm is $(n/r)T(r) + O(n)$ according to Lemma 3.5, where T(r) is the time to solve one subproblem of size r. There is a well known algorithm for the TSP which runs in time $r^2 2^r$ (see Held and

Karp [1962]), giving the time bound stated in the theorem. Since r can grow arbitrarily slowly, the running time can be made as close to linear as desired, but must grow faster than n because $r(n) \to \infty$.

Part (b) follows directly from Lemma 3.5. We are, of course, using the relative error criterion for success of the algorithm. For parts (c) and (d), we will show that under the conditions stated, $n^{-(k-1)/k}Z_n \to_{as} \mu$ in the independent model, after which the remainder of the theorem will follow from Lemma 2.7 and Theorem 2.8.

Let $W_n = n^{-(k-1)/k}Z_n$. Since the normalized tour lengths for the subproblems are i.i.d. random variables with bounded mean (say μ_r) and bounded variance (σ^2_r), the central limit theorem applies, and we have $(n/r)^{1/2}(W_n - \mu_r)/\sigma_r \to_d \mathfrak{N}(0, 1)$. Now,

$$P\{ |W_n - \mu| > \epsilon \} \le P\{ |W_n - \mu_r| > \epsilon/2 \} + P\{ |\mu_r - \mu| > \epsilon/2 \}$$

$$= P\{ |(n/r)^{1/2}(W_n - \mu_r)/\sigma_r)| > (n/r)^{1/2}\epsilon/(2\sigma_r) \} + P\{ |\mu_r - \mu| > \epsilon/2 \}$$

$$= O(\exp(-n\epsilon^2/(8r\sigma^2_r))) + P\{ |\mu_r - \mu| > \epsilon/2 \}.$$

Summing the probability that W_n differs from μ by more than ϵ gives

$$\sum_n P\{ |W_n - \mu| > \epsilon \} \le \sum_n O(\exp(-n\epsilon^2/(8r\sigma^2_r))) + \sum_n P\{ |\mu_r - \mu| > \epsilon/2 \}$$

The first sum on the right hand side is finite for all $\epsilon > 0$ iff $r(n) = o(n/\log n)$, while the second sum is finite if $r(n) \to \infty$, since $\mu_n \to \mu$ (so that all terms are zero from some point on). This proves that $W_n \to_{as} \mu$ in the independent problem model, and the rest of the theorem follows directly. □

One rather puzzling aspect of this theorem is that it seems reasonable to

expect a trade-off between faster growing functions r (and therefore larger solution times) and the success of the approximation. This trade-off is not explicitly mentioned in the theorem. In fact, the algorithm succeeds strongly if r(n) does not grow too fast, but may not do so if r(n) grows too quickly (but still sublinearly). This apparent paradox is related to another trade-off between the rate of convergence of $n^{-(k-1)/k}\chi_n$ to μ, and the number of subproblems which must be solved in order to guarantee almost sure convergence of their average solution value. The conditions necessary for almost sure convergence of the average can be determined as above, but we have no way of knowing what conditions on r would assure almost sure convergence of $r^{-(k-1)/k}\chi_r$, which would be an important part of the proof if $r(n) \neq o(n/\log n)$. The final conclusion is therefore weaker than we would like, but only because we cannot prove more, not necessarily because a stronger conclusion is not true.

If we knew more about the distribution of $n^{-(k-1)/k}\chi_n$, for example that some suitably normalized form were asymptotically normally distributed, then it would be possible to determine bounds on the distribution of the error introduced by subsampling. Here, as mentioned in Section 2.1, the estimate and the exact answer are not independent, but the correlation tends to make the estimate better than it would otherwise be.

This approach to estimating the length of the optimal tour differs from that taken by Golden [1977]. He presents a believable argument but no hint of a proof that for any particular TSP the distribution of tour lengths is approximately of a particular

form known as the Weibull distribution. The optimal tour length is approximately equal to a certain parameter of this distribution, which he shows can be estimated by computing the lengths of several sampled tours. The conclusion that the distribution of tour lengths matches the form of the Weibull distribution is supported by experimental evidence on a number of examples.

Golden's result is of interest not only because it enables one to estimate the optimal tour length for general TSP's, as we were able to do for the ETSP in particular. It is also useful because it suggests a method for analyzing an entire family of approximation algorithms called "λ-opt" heuristics which have been suggested by Lin [1965]. These heuristics have been used for solving a variety of discrete optimization problems (see Croes [1958], Kernighan and Lin [1970, 1973], Lin and Kernighan [1973], Lin [1975], and Nishizeki, Hidenao, and Saito [1977]).

For the general case, λ is an integer parameter of the algorithm which gives some indication of how good the approximation is expected to be. Corresponding to the λ-opt algorithm is the expected computation time of the algorithm, $T_\lambda(n)$, for a problem of size n. The larger the value of λ, the better the solution is expected to be, but the longer the computation time. The user is free to decide on what he considers a fair trade-off between the accuracy achieved and the computing resources consumed, and our job is to give him some basis on which to choose a value of λ once he has settled on this criterion.

The algorithms work in the manner suggested in Section 3.3.2. We randomly

pick m solutions which satisfy some feasibility constraint (the larger the value of λ, the more constrained these solutions are, hence the longer it takes to find them) and simply keep the best of these m solutions as the approximate answer. Throughout the discussion we assume that the number of feasible solutions is much larger than m, which is true for most problems. It is just this fact which makes it imperative that we settle for an approximate answer to the problem.

If there were some way to determine how good this approximation were likely to be as a function of λ and m, we could apply the decision criterion to choose the best values of λ and m. The user would have the ability to effectively produce a table of computation time and accuracy versus λ and m.

The key to the analysis of such algorithms is the observation that for any particular problem instance there is some distribution of values of feasible solutions. Suppose that we consider a fixed problem instance of size n and fix λ and m. Then there is some distribution of values of solutions which are feasible to the constraints associated with the given λ; call this distribution $F_\lambda(x)$. By Lemmas 3.3 and 3.4, the expected relative rank of the solution produced by the algorithm is about $1/(m+1)$, which means that its approximate value is $F_\lambda^{-1}(1/(m+1))$. This is not strictly true, of course, since the actual expected value of the solution is

$$\sum_j P\{ \text{rank} = j \} \cdot (\text{value of } j^{\text{th}} \text{ best solution}).$$

However, if F_λ converges (as $n \to \infty$) to some distribution which is sufficiently smooth, then the conclusion is valid because the relative rank of the approximate solution

converges rapidly to its expected value as $m \to \infty$. In any event, the actual expected value of the solution is not so important as the fact that we have now related the rank of the solution to some value which could be computed if we knew F_λ. An alternative view of the situation is that the values of the m sample solutions are m independent observations on a random variable having the distribution $F_\lambda(x)$. David [1970] has shown that under certain conditions the expected value of the minimum of these random variables is asymptotically $F_\lambda^{-1}(1/(m+1))$.

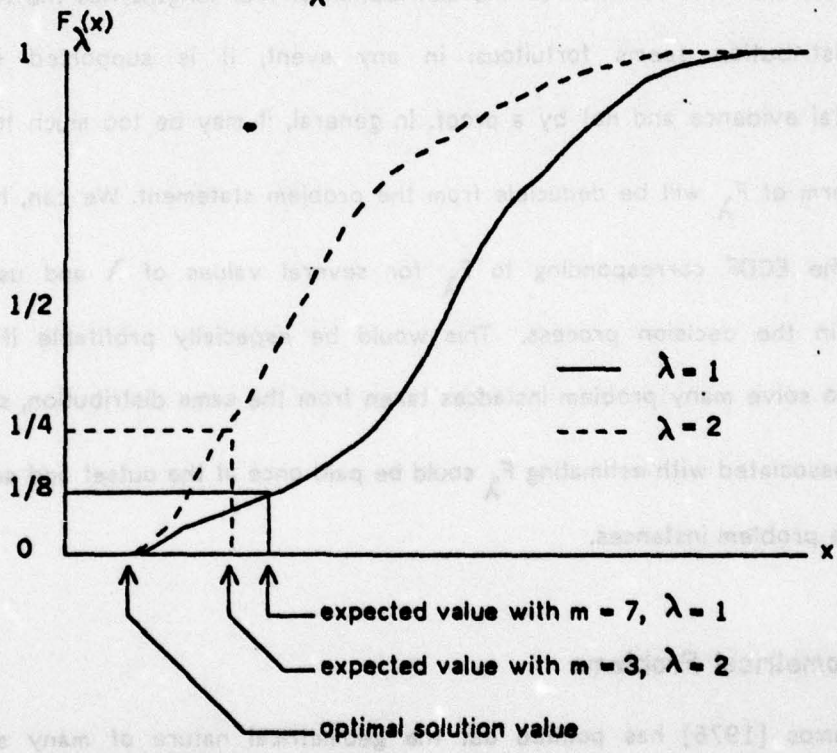


FIGURE 3.10 - Comparison of two approximation algorithms.

Figure 3.10 illustrates how the distributions F_λ might vary with λ for some problem. The figure shows how to compare $\lambda = 1$ with $\lambda = 2$, assuming that the user

has decided to choose λ and m to get the best approximation in total compute time T. Suppose that he has determined that he could find 7 feasible solutions with $\lambda = 1$, or 3 feasible solutions with $\lambda = 2$, in time T. Knowing the distributions $F_\lambda(x)$ he could see that in this case it would be preferable to choose $\lambda = 2$ and $m = 3$. Furthermore, he could approximate the optimal solution value to evaluate the approximation.

It is usually not reasonable to expect that the distributions F_λ will be known. Golden's assertion that for the TSP the distribution of tour lengths has the form of a Weibull distribution seems fortuitous; in any event, it is supported only by experimental evidence and not by a proof. In general, it may be too much to expect that the form of F_λ will be deducible from the problem statement. We can, however, compute the ECDF corresponding to F_λ for several values of λ and use these estimates in the decision process. This would be especially profitable if a user expected to solve many problem instances taken from the same distribution, since the overhead associated with estimating F_λ could be paid once at the outset and amortized over all the problem instances.

3.4.4. Geometrical Problems

Shamos [1976] has pointed out the geometrical nature of many statistics problems and has demonstrated how statistical computations can often be done by fast algorithms for geometry problems because of the geometrical nature of the problems. It is therefore interesting that our statistical techniques are also applicable to

geometry problems. In this section we will see how each of the techniques of Section 3.3 can be used to develop fast expected-time algorithms for problems defined by point sets in Euclidean space. Among these are the closest point problems (see Shamos and Hoey [1975] and Bentley [1976]), for which statistical techniques are especially valuable.

Rabin's [1976] paper on randomized algorithms is important not only because of his probabilistic prime-testing algorithm (particularly useful in light of the current interest in cryptography), but also because it includes a linear expected time algorithm for finding the closest pair of n given points in the plane or higher dimensions. This result holds for any fixed set of points because the method involves a randomized algorithm. The problem is interesting in theory primarily because it was not known to be soluble in $o(n^2)$ time in the worst case until quite recently. Shamos [1975] gave an O(n log n) worst-case algorithm for the planar problem, and Bentley [1976] and later Yuval [1976] discovered different algorithms which run in O(n log n) time in higher dimensions as well.

The algorithms of Shamos and Bentley make no use of the floor function, while Yuval's algorithm does. When we say Rabin's algorithm runs in linear expected time, we mean that the expected number of interpoint distance (IPD) calculations is linear. The entire algorithm runs in linear expected time if constant time computation of the floor function is allowed, which is the case in our model of computation.

The algorithm we propose is extremely simple, in contrast to Rabin's algorithm, and admits of a strikingly simple proof that only a linear number of distance computations are required on the average. Points are assigned to "bins" in a system

of square grids based on the minimum interpoint distance of a small sample, after which only a small number of points within each bin must be checked. The grid method used is due to Yuval [1975a]. The algorithm given here works for points in the plane, but is easily extended to account for higher dimensional problems.

```
procedure closestpair(S);
   set S;
   begin
      integer i, n = |S|;
      real dist;
      randomly sample n/2 interpoint distances, letting
            δ = minimum interpoint distance of sample;
      dist := δ;
      for each point (x,y)∈S do                                        (*)
         for i := 0 to 2 do
            assign (x,y) to bin (⌊(x+iδ)/(3δ)⌋,⌊(y+iδ)/(3δ)⌋) in grid i;
      for i := 0 to 2 do
         for each non-empty bin in grid i do
            dist := min ( dist, minimum IPD in bin ) ;                 (**)
      return(dist)
   end;
```

The algorithm requires that we organize the bins in such a way that all points in the same bin can be quickly retrieved. This is easy to do by sorting without additional distance computations, or by hashing. As mentioned before, this detail is not the main point we wish to make, so it will be ignored in the complexity analysis and we will count only interpoint distance computations.

THEOREM 3.16 -

The procedure "closestpair" finds the distance between the closest pair of points in the set S. The expected number of distance computations required is O(n), where n = |S|.

Proof - It is clear that the closest pair must be no more than δ apart, and it is

easy to show that every pair of points within δ of each other is in the same bin in one of the three grids of mesh size 3δ (see Yuval [1975a, 1976]). Only these points need be checked once δ is determined, and the algorithm does just that.

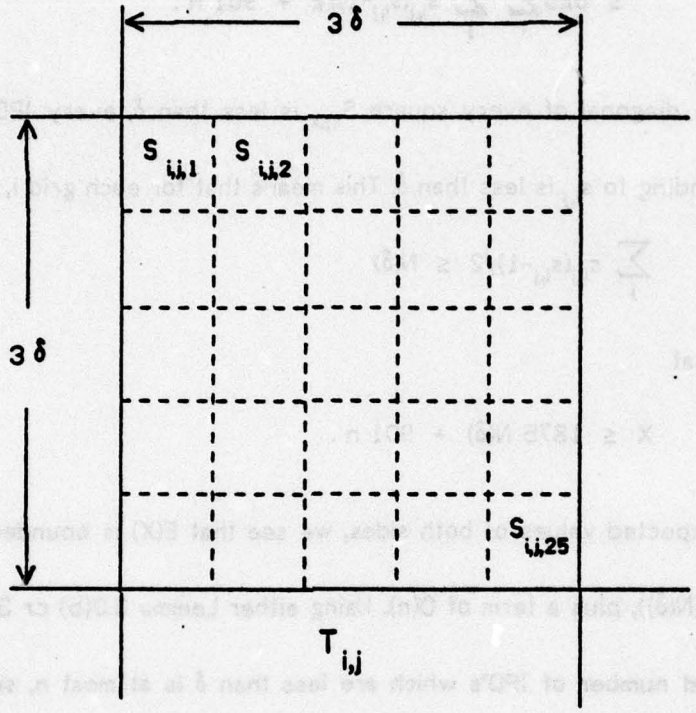


FIGURE 3.11 - A bin in one of the three grids.

Figure 3.11 shows a bin in one of the three grids, say i; call this bin $T_{i,j}$. We consider this bin to be divided into 25 smaller squares $S_{i,j,k}$, $1 \le k \le 25$. Let $t_{i,j}$ be the number of points in $T_{i,j}$; $s_{i,j,k}$ the number of points in $S_{i,j,k}$; and $s_{i,j}$ the maximum of $s_{i,j,k}$ over all subsquares $S_{i,j,k}$. We denote by $N(\delta)$ the total number of IPD's which are less than δ, and by X the total number of IPD's which must be computed by the algorithm.

We have by definition of X that

$$X = \sum_i \sum_j t_{i,j}(t_{i,j}-1)/2 + n/2$$

where i ranges over the three grids and j over all non-empty bins in grid i. We simplify this as follows:

$$X \le \sum_i \sum_j 25s_{i,j}(25s_{i,j}-1)/2 + n/2$$
$$\le 625\sum_i \sum_j s_{i,j}(s_{i,j}-1)/2 + 901\, n .$$

Since the diagonal of every square $S_{i,j,k}$ is less than δ, every IPD in the small square corresponding to $s_{i,j}$ is less than δ. This means that for each grid i,

$$\sum_j s_{i,j}(s_{i,j}-1)/2 \le N(\delta)$$

and therefore, that

$$X \le 1875\, N(\delta) + 901\, n .$$

Taking expected values of both sides, we see that E(X) is bounded above by a constant times $E(N(\delta))$, plus a term of O(n). Using either Lemma 3.3(b) or 3.4(b), we see that the expected number of IPD's which are less than δ is at most n, since the total number of interpoint distances is $\binom{n}{2}$ and δ is the minimum of a random sample of n/2 of them. This proves the theorem. An obvious extension of this argument leads to a proof that a similar algorithm runs in linear expected time in higher dimensions as well. □

An alternative algorithm and proof is suggested by Jon Bentley. An algorithm for finding all points within some distance δ of each other is given by Bentley, Stanat, and Williams [1977] and is shown to require a number of distance calculations which is

$O(n + N(\delta))$. This algorithm could be substituted for the statements labelled (*) through (**) in procedure "closestpair", and only the closest pair of points actually retained. Since the expected value of $N(\delta)$ in this case is $O(n)$ by Lemma 3.3(b) or 3.4(b) as explained above, the alternative algorithm also uses a linear number of distance calculations. This illustrates the value of analyzing an algorithm in terms of the number of outputs produced as well as the number of inputs.

The closest pair problem and the above algorithms provide a good example of the importance of the seemingly trivial Lemmas 3.3 and 3.4. Using these lemmas, one is able to produce a comprehensible proof that the algorithm runs in linear expected time. This should be contrasted with Rabin's [1976] complicated algorithm and his rather involved proof that its running time is linear.

Another simple but interesting problem is that of finding the average of all $\binom{n}{2}$ interpoint distances for n points in Euclidean space. Again we restrict our attention to the planar case for simplicity, although all the results apply for problems in any fixed dimension. Shamos and Yuval [1976] have shown that computing the exact answer requires $\theta(n^2)$ distance computations, but that an approximation which is guaranteed to achieve an absolute error of at most ϵ can be computed in linear time.

A probabilistic approximation algorithm is extremely simple to devise. If we assume only that the points are chosen independently from some distribution for which the distance between two random points has a finite mean μ and finite variance σ^2, then we can prove that an algorithm which samples r(n) independent distances

succeeds weakly or strongly according to whether $r(n) \to \infty$ or $r(n)$ grows faster than $\log n$, respectively. Any distribution which is confined to a bounded region or has sufficiently small tail probabilities will satisfy this condition. In order to prove the main result we need the following lemma.

LEMMA 3.17 -

Let n points be chosen independently from a distribution over the plane for which the expected distance between two points (μ) and the variance of that distance (σ^2) are finite. Let X_n be the average of the $\binom{n}{2}$ interpoint distances defined by those points. Then $X_n \to_{as} \mu$ in the independent model.

Proof - If all the interpoint distances were independent the lemma would be obvious. The difficulty arises because there are only n points and $\binom{n}{2}$ distances, so all the IPD's are not independent.

Consider the complete graph on n points K_n. Each vertex of K_n has degree n-1, so by a theorem of Vizing (see Harary [1969]) the edge-chromatic number of K_n is either n or n-1. Suppose it is n-1 (the other case is similar). By symmetry and the fact that every vertex has n-1 edges incident to it, each of these n-1 color classes has n/2 edges in it. By definition, these edges are independent; that is, they share no common vertices. Their lengths are therefore independent, so the expected value of the average of the IPD's within each color class is μ, and the variance is $2\sigma^2/n$. Furthermore, by the central limit theorem we know that the limiting distribution of the average is normal.

Let the average IPD within color class k be $Y_{n,k}$ for $1 \le k \le n-1$. The average IPD for the entire set of points is then $X_n = (1/(n-1))\sum_k Y_{n,k}$. We easily find that

$$P\{ |X_n - \mu| > \epsilon \} = P\{|(1/(n-1))\sum_k Y_{n,k} - \mu| > \epsilon \}$$

$$\le \sum_k P\{ |Y_{n,k} - \mu| > \epsilon \}$$

$$= O(n \cdot \exp(-\epsilon^2 n/(4\sigma^2)))$$

which means that $\sum_n P\{ |X_n - \mu| > \epsilon \}$ is finite for all $\epsilon > 0$, proving the lemma. □

It is now clear how a sampling algorithm will work. We first randomly choose a sample of r(n) points from the original n, then another sample of r(n) points from those remaining (we assume that $r(n) \le n/2$). The points of the first sample are matched with those of the second and the r(n) IPD's so determined are averaged. This is the estimate of the overall average distance. It is clear that the time required by this algorithm is O(r(n)). The next theorem tells us how well the algorithm performs.

THEOREM 3.18 -

(a) If $r(n) \to \infty$ then the above algorithm succeeds weakly.

(b) If r(n) grows faster than log n then the above algorithm succeeds strongly in the independent model.

Proof - Let Z_n be the estimate produced by the algorithm. In light of Lemma 3.17, it is only necessary to show that Z_n converges to μ in the appropriate ways under the conditions of the theorem. This is an easy exercise similar to many other proofs in this chapter and the previous one because the r(n) IPD's are independent. Notice that the algorithm succeeds in these modes even using the absolute error criterion. □

The first two examples of this section have been almost trivial applications of techniques developed in Section 3.3. As another illustration of the utility of those methods, we move on to another rather easily stated problem, called nearest neighbor searching, which apparently has no easy solutions. Given n points in Euclidean space (again, consider only the planar case now) we are asked to preprocess the points in linear expected time so that given a new point we can determine which point of the original n is closest to it in constant expected time. There are (complicated) structures available for solving the problem with O(n log n) preprocessing time and O(log n) query time in the worst case (Lipton and Tarjan [1977]), but one might expect that on the average we can do better. Although we will restrict our attention to this apparently simple problem, the techniques used also apply to other closest point problems. In particular, we will briefly describe how it is possible to construct the Voronoi diagram in linear expected time under conditions similar to those of Theorem 3.20, which means that the planar minimum spanning tree, all nearest neighbors, and Delaunay triangulation problems, among others, can be solved in linear expected time. (The Voronoi polygon of a point of the set is the locus of all points which are closer to that particular point than to any other point of the set, and the Voronoi diagram is the collection of the Voronoi polygons of all points of the set; see Shamos and Hoey [1975].)

We first consider the problem of nearest neighbor searching, often called the post office problem, where the points (both the original n points and the query point) are chosen independently from a uniform distribution over the unit square. The idea of the preprocessing step is to assign each point to a small square ("bin" or "cell") of area C/n, so that the expected number of points in each bin is C. This is easily done

by creating an array of size $(n/C)^{1/2}$ by $(n/C)^{1/2}$ which holds pointers to the lists of points in each bin.

When a query point arrives, we search the bin it would be assigned to. If it is empty, then we start searching, in a spiral-like pattern, the cells surrounding it, until a point is found. Once we have one point, we are guaranteed that there is no need to search any bin which does not intersect the circle of radius equal to the distance to the first point found and centered at the query point. Figure 3.12 shows how the spiral search might proceed for the query point in the middle of the circle. Once the point in the northeast neighbor is found, only bins intersecting the circle must be searched. These are marked with an x in Figure 3.12. In order to make this test easy, we suppose that all bins which lie within that distance of the query point in either coordinate are examined, making the number of cell accesses slightly larger than necessary but simplifying the specification of how the appropriate bins are to be found.

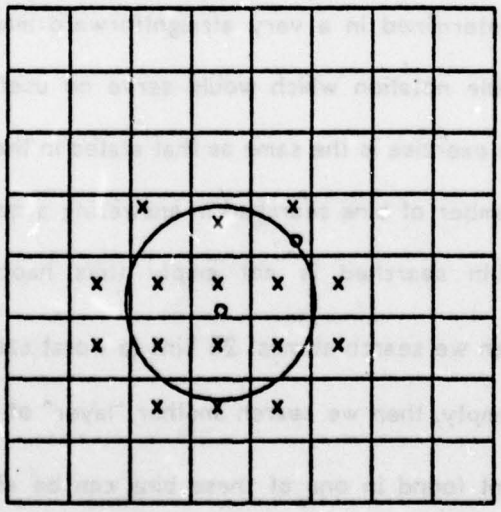


FIGURE 3.12 - Spiral search for a nearest neighbor.

It is clear that preprocessing, which consists of assigning each point to the appropriate bin, can be accomplished in linear time if constant-time computation of the floor function is allowed. As before, this is possible in our model of computation. The following lemma shows that spiral search is a solution to the problem we originally posed if the points are uniformly distributed.

LEMMA 3.19 -

If n points are chosen independently from a uniform distribution over the unit square, then the nearest neighbor of a query point can be found in constant expected time using the spiral search.

Proof - We merely need to determine the expected number of bins which must be examined by the algorithm. The effects of dependence among the numbers of points in each bin will become negligible as n grows, so we make the simplification that the expected number of distance computations which are made is approximately equal to C times the number of bins searched. The exact expected number of distance computations can be determined in a very straightforward manner, but only at the expense of unmanageable notation which would serve no useful purpose here. The conclusion from such an exercise is the same as that stated in the theorem.

Let Q be the number of bins searched in answering a query when there are n points. If the first bin searched is not empty (this happens with probability $1-(1-C/n)^n \rightarrow 1-e^{-C}$) then we search at most 25 bins (a worst case bound which cannot be achieved). If it is empty, then we search another "layer" of up to 8 bins centered at the first one; a point found in one of these bins can be at most $2^{3/2}E$ from the query point, where E is the length of each side of a cell, so the total number of bins

searched is at most $\left(2\lceil 2^{3/2}\rceil+1\right)^2$, or 49. In general, if the first point is found in the k^{th} layer of the spiral search, we must examine at most $\left(2\lceil k2^{1/2}\rceil+1\right)^2 \le 8k^2 + 23k + 16$ cells. Summing the probability that the first point is found in layer k, times the number of bins examined if that occurs, gives

$$E(Q) \le 25\,(1 - e^{-C}) + \sum_{k\ge 2}(8k^2 + 23k + 16)\, e^{-(2k-3)^2C}\,(1 - e^{-8(k-1)C})$$

$$< 25 + \sum_{k\ge 2}(8k^2 + 23k + 16)\, e^{-(2k-3)^2C}$$

This last sum is bounded above by a constant, so E(Q) is bounded by a constant. Therefore, the expected total number of points examined is bounded by a constant independent of n. □

The extension of this result to unknown distributions is a bit tricky. If we proceed for such a distribution as though it were uniform over some bounded region (i.e., we use the ECDF with no intermediate sample points but only the extrema in each coordinate), a query can still be answered in constant expected time if the actual underlying distribution satisfies a condition similar to but more restrictive than a Lipschitz condition.

THEOREM 3.20 -

Let n points be chosen independently from the distribution F(x,y) over a bounded region of the plane, where F satisfies the condition that there exist constants $0 < C_1 \le C_2 < \infty$ such that for any region of area A, the probability assigned to A by F lies between C_1A and C_2A. (Alternatively, F has a density with respect

to Lebesgue measure which is bounded above and bounded away from zero.) Then the same algorithm which was used in the uniform case answers a query in constant expected time.

Proof - The proof of Lemma 3.19 is easily modified to show that the expected number of distance computations is at most

$$C_2 \left(25 + \sum_{k \geq 2} (8k^2 + 23k + 16)\, e^{-(2k-3)^2 C_1} \right)$$

which is bounded by a constant independent of n. The only difficulties which arise are caused by the possibly irregular boundary of the region from which the points are chosen. Since the region and distribution are fixed and independent of n, it is straightforward to show that even while the bound above holds for points "sufficiently far from the boundary", the only modification required for points "near the boundary" is that the number of comparisons may be larger by a constant factor. □

This algorithm is equivalent to one which bases its estimate of the distribution on only the minimum and maximum coordinate values in each dimension and assumes the points are uniformly distributed in the region they define. This is precisely the two dimensional analogue of the ECDF of Section 3.3.4 based on no intermediate sample points. As pointed out there, in practice it is usually wise to make a more reasonable estimate of the distribution by sampling several (but a constant) number of points to divide the entire region into many smaller regions, within which uniformity is assumed. The trade-off involved in choosing the sample size is that the number of distance calculations may be reduced, but the cost of looking up the bin in which a point is stored is increased. Whether a more sophisticated estimate of the underlying distribution will be beneficial also depends to a large extent on how irregular that

distribution really is, something which can be decided from an initial sample of the data as suggested for Binsort.

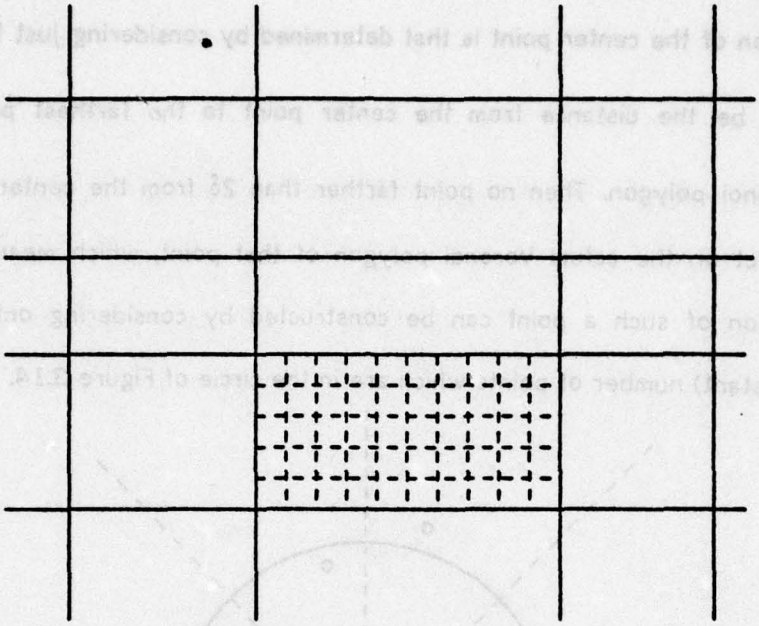


FIGURE 3.13 - A grid system for nearest neighbor searching.

In case intermediate sample points are taken, there are many possible methods available for defining a grid and modifying the spiral search strategy. One possibility is to divide each coarse grid region into a number of smaller (approximately) square regions proportional to the number of points which actually fall in that rectangle, as suggested by Figure 3.13.

The problem of constructing the Voronoi diagram of a planar point set is somewhat more delicate[6]. The basic idea is to search all cells in a relatively small

---

6 . Enough so that numerous discussions with Jon Bentley and Mike Shamos were necessary to iron out several bugs which plagued the original algorithm. The idea that cell techniques could help in closest point problems is due to Jon Bentley.

neighborhood of each point (at most log n cells) in a spiral-like fashion until at least one point is found in each of the eight octants shown in Figure 3.14. The "tentative" Voronoi polygon of the center point is that determined by considering just those eight points. Let δ be the distance from the center point to the farthest point of its tentative Voronoi polygon. Then no point farther than 2δ from the center point can have any affect on the actual Voronoi polygon of that point, which means that the Voronoi polygon of such a point can be constructed by considering only the few (expected constant) number of points which are in the circle of Figure 3.14.

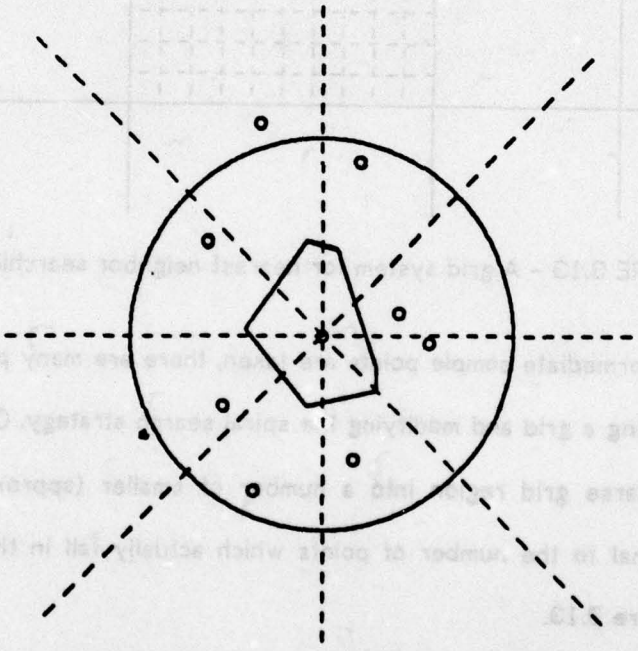


FIGURE 3.14 - A tentative Voronoi polygon and its sphere of influence.

In case there is at least one point found in each octant before the log n cells are searched, the point is called a "closed" point. Fortunately, it can be shown that all but $O((n \log n)^{1/2})$ points are closed, and that for each closed point the work to

construct the desired Voronoi polygon is expected to be constant. All other points, most of which are near the boundary of the region from which the points are drawn, are called "open" points. Since there are $O((n \log n)^{1/2})$ of these and each is identified in log n steps, the total work required to identify the open points is $O(n^{1/2}\log^{3/2}n)$.

Once the Voronoi polygons of the closed points are constructed and the open points are identified, all of which takes linear expected time, the diagram is completed by applying the O(n log n) Voronoi diagram algorithm of Shamos and Hoey [1975] to the set of open points plus all closed points which share an edge with some open point. The expected size of this set is $O((n \log n)^{1/2})$, so the expected time required by the O(n log n) worst-case algorithm is $O(n^{1/2}\log^{3/2}n)$. A final pass taking at most linear time is then done to remove from this diagram all edges shared by two closed points. The superposition of the remaining diagrams, one from the closed points and the other a modified version of that from the open points and their closed neighbors, is the Voronoi diagram of the entire set. A more complete description and analysis of the algorithm is given by Bentley, Weide, and Yao [1978].

As a final example of a geometrical problem which falls to statistical techniques we offer the case of finding the convex hull of a set of points in the plane. The linear expected time algorithm of Bentley and Shamos [1978] cleverly makes use of a preprocessing randomization step. The main idea of the algorithm is the divide-and-conquer strategy mentioned briefly in Section 3.3.4. In the standard recurrence

$$T(n) = 2\,T(n/2) + D(n) + M(n)$$

the solution if $T(n) = O(n)$ if $D(n)$ and $M(n)$ are $O(n^h)$ for some $h < 1$.

In the problem of the convex hull, the expected number of points on the hull of n points is $O(n^h)$ for some $h < 1$ for many distributions of points, among these all sets of points with independently distributed coordinates. The Bentley-Shamos algorithm randomly divides a problem of size n into two problems of size n/2, finds the convex hull of the points in each subproblem recursively, then marries the solutions together to find the convex hull of the original point set. The marriage step can be shown to be linear in the total number of points on the two hulls being combined, so $M(n) = O(n^h)$ for $h < 1$.

The difficulty arises because the obvious method for doing a random division of n points into two sets of n/2 each requires time linear in the number of points. If that division method were used, the total time required by the algorithm would be O(n log n). Rather than creating subproblems as necessary, which is the usual method in divide-and-conquer algorithms, the division process is consolidated into a preprocessing step in which all the subproblems are essentially created at once. The points are stored in an array, which is initially randomized (in linear time). After this, when a new subproblem is to be solved it is specified by two indices into the array, and all points between those two positions are included in the subproblem. This makes D(n) constant, and adds a term O(n) to the total running time for the randomization step, so the entire algorithm uses only linear expected time. Notice that the randomization step guarantees that the subproblems are random and that the points of any subproblem have the same distribution as the original set of points.

This algorithm provides a very nice example of the non-trivial nature of the

randomization technique. Other applications include finding convex hulls in 3-space, finding maxima of vectors in k-space, and linear programming in two or three variables.

3.5. Conclusions

In this chapter we have seen how some very simple ideas from statistics lead naturally to design and analysis techniques for efficient algorithms. Many of the results reported here are of practical importance, especially those related to sorting and selection, because algorithms for these problems are at the heart of algorithms for many other problems. As a simple example we note that Graham's [1972] convex hull algorithm can be implemented to run in linear expected time for a very wide class of distributions if the sorting step uses Binsort.

Nevertheless, it is really the techniques at hand and not just the results themselves which make statistical methods useful in designing and analyzing discrete algorithms. We have seen at least two examples of the use of each of the techniques described in Section 3.3, and other problems for which the methods apply have been suggested. There is ample evidence to support the assertion that a more comprehensive review of the tools used in statistics will reveal other transferable technology.

4. Order Statistics

The statistical tools of Chapter 3 are generally based on well-known concepts, many of which have been previously applied by computer scientists to algorithmic problems. In contrast, the field of statistics that, among other things, deals with the distribution of extreme values of random variables (called order statistics) is less widely understood and usually applied only by statisticians themselves. We propose to show here that many simple results from this interesting area can help in the analysis of probabilistic approximation algorithms and parallel algorithms.

Section 4.1 contains a summary of some important theorems from order statistics and a lemma which helps in the computation of the asymptotic distribution of order statistics. Several examples of the application of these results to algorithmic problems are introduced in Section 4.2. Finally, some conclusions are presented in Section 4.3.

4.1. Expected Values and Asymptotic Distributions

If the n random variables $\{X_i\}$ are sorted into increasing order by observed values, then we call the k^{th} smallest one the k^{th} order statistic of the sample, and denote it by $X_{(k)}$ or, if the value of n is not apparent from the context, by $X_{k:n}$. The

subject of order statistics deals with the distributions, expectations, and other statistical properties of the random variables $X_{(k)}$. Our most important references for this chapter are David [1970] and Gumbel [1958], which contain many interesting and important theorems and applications dealing with order statistics.

In general, the random variables X_n need not be independent and identically distributed, but throughout this chapter we assume that this condition is satisfied. Proofs are much easier in this framework, and even if some form of dependence were allowed it is not clear that real applications would come closer to satisfying whatever condition we chose than they do to satisfying independence.

Beginning with the assumption that we have a sample of n observations on a random variable X with distribution F, we can prove several well-known lemmas which provide the foundation for further study.

LEMMA 4.1 -

$$F_{(k)} = F_{k:n}(x) = P\{ X_{(k)} \le x \} = \sum_{k \le i \le n} \binom{n}{i} F^i(x) (1-F(x))^{n-i}$$

Proof - The probability that $X_{(k)}$ is less than or equal to x is just the probability that at least k of the n observations are less than or equal to x. Term i of the sum is the probability that exactly i of the observations do not exceed x, and we have summed these probabilities to obtain the desired expression for the distribution of $X_{(k)}$. □

LEMMA 4.2 -

Let $\mu_{(k)} = \mu_{k:n} = E(X_{(k)})$. We have

$$\mu_{(k)} = n \binom{n-1}{k-1} \int x \, F^{k-1}(x) \, (1-F(x))^{n-k} \, dF(x)$$

Proof - This follows from Lemma 4.1 and the definition of $E(X_{(k)})$. See David [1970], page 25. □

There are many relationships among expected values of order statistics which are closely analogous to certain relationships among binomial coefficients, which is no great surprise. The following exemplifies this situation, and will be important in Section 4.2.3.

LEMMA 4.3 -

$$(n-k)\mu_{k:n} + k\mu_{k+1:n} = n\mu_{k:n-1}$$

Proof - See David [1970], page 37. □

For reasons which are apparent as soon as one tries to compute distributions and expected values of order statistics for various distributions F, two favorite distributions are the exponential and uniform. If the observations are from an exponential distribution with unit mean, then $\mu_{k:n} = H_n - H_{n-k}$, where H_n is the n^{th} harmonic number $\sum_{1 \le i \le n} (1/i)$. If the underlying distribution is uniform between 0 and 1, then $\mu_{k:n} = k/(n+1)$. For other important distributions, including the normal, no closed forms are generally known for $\mu_{k:n}$ or for $F_{k:n}(x)$.

It is possible nevertheless to obtain asymptotic distributions for order statistics which apply to a wide range of underlying distributions. Perhaps the most important order statistics are the minimum $X_{(1)}$ and maximum $X_{(n)}$ of a sample of size n, for which limiting distributions have long been known (see David [1970]). In the following lemma,

we prove a simple result for the maximum $X_{(n)}$ which leads to the development of the three so-called "extreme value distributions".

LEMMA 4.4 -

Let $z \geq 0$ be fixed and let $n \to \infty$.

(a) If there exists $\epsilon > 0$ such that $F(F^{-1}(y)) = y$ for all $0 < y < \epsilon$, then

$$P\{ X_{1:n} \leq F^{-1}(z/n) \} \to 1 - e^{-z}.$$

(b) If there exists $\epsilon > 0$ such that $F(F^{-1}(y)) = y$ for all $1-\epsilon < y < 1$,

then $P\{ X_{n:n} \leq F^{-1}(1-(z/n)) \} \to e^{-z}$.

Proof - We prove only (b), since (a) is virtually identical. Note that $P\{ X_{n:n} \leq x \} = P\{ X \leq x \}^n$. By hypothesis of the lemma, for all sufficiently large n we have $P\{ X \leq F^{-1}(1-(z/n)) \} = 1-(z/n)$. The lemma is proved by combining these relationships with the fact that $(1-(z/n))^n \to e^{-z}$ for every fixed z as $n \to \infty$. □

There are obviously similar results for other order statistics, but the distributions of the maximum and minimum of n i.i.d. random variables are the most important for our purposes and, fortunately, the easiest to express. Using Lemma 4.4, we can find the asymptotic distributions of the maximum and minimum for an arbitrary population by using an asymptotic expansion for F^{-1} if one is available. For the exponential distribution with unit mean, $F(x) = 1-e^{-x}$ for $x \geq 0$, so $F^{-1}(y) = -\ln(1-y)$. This means that

$$P\{ X_{(n)} \leq F^{-1}(1-(z/n)) \} = P\{ X_{(n)} \leq -\ln(z/n) \}$$

$$= P\{ X_{(n)} - \ln n \leq -\ln z \}.$$

Substituting x for - ln z in this expression and in the limit of Lemma 4.4 gives

$$P\{ X_{(n)} - \ln n \le x \} \to e^{-e^{-x}}$$

which applies for every fixed value of x as $n \to \infty$.

A similar exercise for the normal distribution is instructive. We have $F^{-1}(y) = (-2 \ln(1-y))^{1/2}(1+o(1))$ as $y \to 1$, from which we can easily show that

$$P\{ (2 \ln n)^{1/2} (X_{(n)} - (2 \ln n)^{1/2}) \le x \} \to e^{-e^{-x}}.$$

This means that a suitably normalized form of the maximum of n normally distributed random variables has the same limiting distribution as a suitably normalized form of the maximum of n exponentially distributed random variables.

It turns out that the probability distribution $e^{-e^{-x}}$ is so important in order statistics that it is often called the extreme value distribution, even though other limiting distributions are possible. If we denote this distribution by η, then the last statements above translate to

$$X_{(n)} - \ln n \to_d \eta \qquad \text{(exponential)}$$

$$(2 \ln n)^{1/2} (X_{(n)} - (2 \ln n)^{1/2}) \to_d \eta \qquad \text{(normal)}$$

While it may not be true in general that the moments of some standardized form of the extreme converge to the moments of the limiting distribution, it is nevertheless instructive to examine the moments of η in order to get an idea of the behavior of the extreme value distribution. Also, it is true that the moments converge if they are finite for sufficiently large n (see Pickands [1968]). It can be shown that

the mean of a random variable having the extreme value distribution is $\gamma = 0.5772...$ (Euler's constant), so that for the exponential distribution we have $E(X_{(n)} - \ln n) \to \gamma$. This agrees with the result cited earlier that $E(X_{(n)}) = H_n$, since $H_n - \ln n \to \gamma$ by definition of γ.

4.2. Examples

In this section we will see how results similar to those presented in Section 4.1 can be used in the analysis of greedy algorithms for optimization problems, parallel algorithms for asynchronous multiprocessors (see Baudet [1978]), and problem decomposition for asynchronous multiprocessors. While many of these results are very simple if ideas from order statistics are used, similar results have previously been obtained only for certain very special distributions. For example, the assignment problem with uniformly distributed matrix entries has been examined many times, but only Borovkov [1962] has generalized the conclusions to arbitrary distributions. Here we will see how restrictive assumptions can be avoided in such analyses for problems ranging from the well-studied assignment problem to scheduling tasks on a multiprocessor.

4.2.1. Greedy Algorithms

The results of this section, which are presented specifically for the traveling salesman problem, generalize almost without modification to the assignment problem, the minimum spanning tree problem, the minimum weighted matching problem, and many similar problems defined on weighted graphs. We assume throughout that some

objective function defined by edge weights is being minimized subject to some constraints, although the changes necessary to achieve similar results for maximization problems are clear.

For the purpose of having a concrete problem in mind, we adopt the traveling salesman problem as our example. Given a complete undirected graph with weighted edges, we seek a closed path which visits each of the n vertices exactly once and minimizes the sum of the weights of the edges traversed[1]. This problem is known to be NP-complete, and the best known guaranteed approximation algorithm (Christofides [1976]) always gets within a factor of 3/2 of the optimal solution and runs in $O(n^3)$ time if the edge weights satisfy the triangle inequality. Otherwise, no guaranteed approximation algorithm is known which runs in polynomial time.

As usual, it is possible to produce a tour which is usually close to optimal in practice but is not guaranteed to be so. The simplest algorithm displaying such behavior is the greedy algorithm, which runs in $O(n^2)$ time.

```
procedure greedy(G);
   graph G;
   begin
      integer i,j;
      real cost = 0;
      mark each vertex as not visited;
      randomly pick some vertex of G as starting vertex and visit it;
      for i := 2 to n do
         begin
            find a vertex j not already visited such that
               the weight of the edge from the last vertex
               visited to j is minimum;
            cost := cost + weight of edge from last vertex
                  visited to j;
            visit vertex j
         end;
      return(cost + weight of edge from last vertex
                  visited to starting vertex)
```

---

1. Sometimes we say "length" rather than "weight" for variety.

end;

The primary virtue of the greedy algorithm from the standpoint of analysis is that it is purely sequential and all decisions are local. The weight of the first edge of the tour depends only on the weights of the edges incident to the starting vertex, and once that edge is chosen we can forget about the starting vertex and all its edges. This property is maintained throughout the operation of the procedure, except at the very end, where we are stuck with the lone remaining edge, however large its weight may be. The important point here is that, except for the last edge of the tour, the weights of the edges chosen are independent of each other. The independence property is a characteristic of greedy algorithms for the other problems mentioned above and explains why they can be analyzed in a similar fashion.

Although it is possible to obtain interesting results for distributions satisfying less restrictive conditions, we will concentrate on edge weight distributions which we call fixed-cost distributions, or FC distributions. An FC distribution F has three properties: (1) There exists some $A = \sup \{ x: F(x) = 0 \} > 0$; (2) F can be expanded in the form $F(x) = F(A) + c(x-A)^{\delta}(1+o(1))$ as $x \to A^{+}$, with $\delta > 0$; and (3) F has finite mean and variance. Intuitively, the condition that $A > 0$ means that we may consider each edge to have associated with it a fixed cost of traversal A and some random non-negative variable cost. The total cost of choosing an edge for the tour is the sum of the fixed and variable costs. Such a model is reasonable in many practical circumstances where allowing edges with zero or negative weights would defy reality.

The analysis of the greedy algorithm assumes that each edge weight is an

independent observation on a random variable having an FC distribution. We begin with the following lemma which bounds the length of the optimal tour and relates the length of the greedy tour to the distribution of edge weights.

LEMMA 4.5 -

Let T_n be the length of the optimal traveling salesman tour for a complete graph on n vertices with edge weights drawn from the FC distribution F; let Y_n be the length of the greedy tour for the graph; and let $X_{k:n}$ be the k^{th} smallest of n random variables X having the distribution F. If Z_n is the weight of the last edge included by the greedy algorithm, then

$$nA \leq T_n \leq Y_n = \sum_{1 \leq i \leq n-1} X_{1:i} + Z_n$$

Proof - Since each edge has weight at least A, and every tour has exactly n edges, the first inequality follows. By definition, T_n is the weight of an optimal tour, so it is at most equal to the weight of the greedy tour. Finally, the operation of the greedy algorithm assures us that the weight of the i^{th} edge included in the tour is the minimum of the n-i edges from the current vertex to the remaining unvisited vertices. These edge weights are independent of the weights of all other edges, not being conditioned by previous choices because none is incident to a vertex already examined. Only when i = n, and no such edges remain, are we forced to choose an edge which has been previously checked. This edge has weight Z_n in the terminology of the lemma. □

Lemma 4.5 is the basis for the analysis of the greedy algorithm. The intuitive meaning of the following theorem is that essentially all of the edges of the optimal tour have weight very close to A. More importantly, the same is true of the greedy tour.

THEOREM 4.6 -

> Under the conditions of Lemma 4.5, we have $T_n/n \rightarrow_{as} A$ in the independent model. Furthermore, $Y_n/n \rightarrow_{as} A$, so the greedy algorithm succeeds strongly in the independent model using the relative error criterion.

Proof - We show that $Y_n/n \rightarrow_{as} A$ in the independent model, from which it follows by Lemma 4.5 that $T_n/n \rightarrow_{as} A$. Then, using Lemma 2.7 (the relative error lemma), we can see that the greedy algorithm succeeds strongly in the independent problem model.

The random variable Z_n is conditioned only by the fact that it is not the smallest of the n-1 edges emanating from the starting vertex, an effect which diminishes as n increases. Thus, Z_n essentially has the same distribution as X (namely F) for large values of n; that is, it is simply a typical edge of the graph, not one with a particularly large weight. Since F has finite mean and variance, it is easily demonstrated that $P\{ |Z_n/n| > \epsilon \} = O(1/n^2)$, so $Z_n/n \rightarrow_{as} 0$. We can therefore continue with the proof as though Y_n did not include the weight of the last edge, since its contribution to the total tour length becomes negligible as n increases.

We determined previously that the random variables $X_{1:i}$ in the sum are independent. This allows us to conclude that

$$E(Y_n/n) = (1/n) \sum_{1 \le i \le n-1} E(X_{1:i})$$

$$D(Y_n/n) = (1/n) \sum_{1 \le i \le n-1} D(X_{1:i}) .$$

The main line of the proof will be to find $E(X_{1:i})$ and $D(X_{1:i})$ and then to use Chebychev's inequality and the Borel-Cantelli Lemma to prove the almost sure convergence of Y_n/n.

According to the definition of a FC distribution, we can write $F(x)$ as $F(A) + c(x-A)^{\delta}(1+o(1))$ as $x \to A^+$, where $\delta > 0$. We assume that $F(A) = 0$, for supposing otherwise only reduces the expected value and variance of $X_{1:i}$, and the theorem will still follow from this argument. Expressing x as a function of $F(x)$ then gives

$$F^{-1}(y) = A + (y/c)^{1/\delta} (1 + o(1)) \quad \text{as} \quad y \to 0 .$$

Applying Lemma 4.4 gives

$$P\{ (ci)^{1/\delta} (X_{1:i} - A) \le x \} \to 1 - e^{-x^{\delta}} .$$

Now, using a theorem of Pickands [1968] which assures us that the first two moments of the random variable $(ci)^{1/\delta}(X_{1:i}-A)$ converge to the moments of the limiting distribution (because the moments are finite by our conditions on F), we have

$$E(X_{1:i}) = A + O(i^{-1/\delta})$$

$$D(X_{1:i}) = O(i^{-2/\delta})$$

because the moments of the limiting distribution are constant. From this we can compute the expectation and variance of Y_n/n as

$$E(Y_n/n) = A + O(1/n) \qquad \delta < 1$$

$$= A + O(\log n/n) \qquad \delta = 1$$

$$= A + O(n^{-1/\delta}) \qquad \delta > 1$$

$$D(Y_n/n) = O(1/n^2) \qquad \delta < 2$$

$$= O(\log n/n^2) \qquad \delta = 2$$

$$= O(n^{-1-2/\delta}) \qquad \delta > 2$$

Now we can compute $\sum_n P\{ |Y_n/n - A| > \epsilon \}$ and apply the Borel-Cantelli Lemma. Using the Chebychev inequality,

$$P\{ |Y_n/n - A| > \epsilon \} = P\{ |Y_n - E(Y_n) + O(n^{-1/\delta})| > \epsilon \}$$

$$\leq P\{ |(Y_n/n - E(Y_n/n)) / (D(Y_n/n))^{1/2}| > \epsilon/(2(D(Y_n/n))^{1/2}) \}$$

$$+ P\{ O(n^{-1/\delta}) > \epsilon/2 \}$$

$$\leq 4\, D(Y_n/n) / \epsilon^2 + P\{ O(n^{-1/\delta}) > \epsilon/2 \}.$$

The first term is $O(1/n^2)$, $O(\log n/n^2)$, or $O(n^{-1-2/\delta})$, depending on δ, while the second term is zero for all sufficiently large n. The sum $\sum_n P\{ |Y_n/n - A| > \epsilon \}$ is therefore finite, which proves that $Y_n/n \rightarrow_{as} A$ in the independent problem model. □

By now it is obvious that the restriction to FC distributions is essential for this proof of the almost sure success of the greedy algorithm, because nA is a lower bound on the optimal tour length. If zero or negative weight edges were possible, then the optimal tour could have weight zero, in which case "relative error" would be

meaningless. If the actual optimal tour length is almost always non-zero, however, then we can sometimes extend the analysis above to distributions which do not satisfy the FC criteria. The upper bound argument applies virtually without modification to normally distributed edge weights with zero mean and unit variance, for example, and we can find a lower bound (the sum of the n smallest edges of the graph) and an upper bound (the weight of a greedy tour), both of which converge in probability to the same constant; in particular, $T_n/(n(2 \ln n)^{1/2}) \rightarrow_{pr} -1$.

Borovkov [1962] has already proved convergence in probability, though, and we want almost sure convergence. If we replace the relatively weak Chebychev inequality by a much stronger one based on the asymptotic normality of the greedy tour length (proved by application of the central limit theorem) then we can prove that the same upper bound holds almost surely in the independent model. However, this method apparently fails if we try to prove that the same lower bound holds almost surely.

Another very interesting approach to these problems is based on the theory of random graphs. The ideas behind Theorems 4.7 and 4.8 are due to T. Nishizeki. The first theorem applies to graphs which are not complete and to arbitrary edge weight distributions.

THEOREM 4.7 -

> Let G_n be a random weighted graph on n vertices in which each edge is independently present with probability p, and each extant edge has a weight chosen from the distribution F. Let [a,b] be any interval to which F assigns positive probability. Then there exists

a constant c such that if $p > c \log n/n$, then G_n contains a tour with weight in the interval [na,nb], almost surely in the independent model.

Proof - For any interval [a,b] to which F assigns positive probability, consider the subgraph of G_n induced by the edges with weight in the interval [a,b]. This is a random graph in which each edge is present with probability $p(F(b)-F(a^-))$, where p is the probability of an edge appearing in G_n.

By a theorem of Posa [1976], there is a constant C such that if the probability of each edge being present exceeds $C \log n/n$, then G_n has a Hamiltonian circuit, almost surely in the independent model. If we set $c = C/(F(b)-F(a^-))$, which is finite since $F(b)-F(a^-) > 0$, then if $p > c \log n/n$, the subgraph induced by the edges with weight in the interval [a,b] almost surely has a Hamiltonian circuit. That circuit is a tour by definition, and has weight in the interval [na,nb]. □

A general form of Theorem 4.7 is useful for bounding the behavior of the optimal solution to a graph optimization problem. Suppose that we have a complete graph with edge weights chosen independently from a distribution F. The problem is to minimize the sum of the weights of the edges in a subgraph over all subgraphs which satisfy certain structural constraints. Theorem 4.8 relates the value of the solution to such an optimization problem to the existence of a subgraph satisfying the constraints in a random graph. For example, it relates the length of the minimum traveling salesman tour to the existence of a Hamiltonian circuit, the weight of the minimum spanning tree to connectedness, or the value of a k-clique to the existence of one.

THEOREM 4.8 -

Let G_n be a random labelled graph in which each edge is present independently with probability p; let C_n be a set of labelled graphs on n vertices, each of which has k_n edges; and let p_n be such that if $p \geq p_n$, then G_n contains at least one of the graphs of C_n as a subgraph, almost surely in the independent model. Given a complete graph on n vertices with edge weights chosen independently from the distribution F, let X_n be the sum of the weights of the edges of a graph in C_n for which that sum is minimum. Then

$$X_n \leq k_n F^{-1}(p_n)$$

almost surely in the independent problem model.

Proof - Consider the subgraph of the complete weighted graph consisting of all n vertices and just those edges of weight at most $F^{-1}(p_n)$. This is just a random graph G_n with $p = F(F^{-1}(p_n)) \geq p_n$. By hypothesis of the theorem, G_n contains a member of C_n almost surely, and because each edge of that subgraph has weight at most $F^{-1}(p_n)$ and it has k_n edges, the weight of that subgraph is at most $k_n F^{-1}(p_n)$. □

Under the conditions of Lemma 4.5 and Theorem 4.6, we now have an alternative proof that $T_n/n \to_{as} A$ for FC distributions. Theorems 4.7 and 4.8 provide the upper bound and nA is a lower bound on the tour length, but this works only for

FC distributions. A method of proving almost sure lower bounds in the manner of Theorem 4.8 would be very useful because it might facilitate proofs of the almost sure convergence of optimal solution values for certain graph optimization problems. However, Theorems 4.7 and 4.8 say nothing about the length of a greedy tour, and it seems that only less elegant arguments such as the proof of Theorem 4.6 are available for analyzing greedy algorithms.

The results of this section are considerably stronger than those reported recently by Lueker [1978]. He shows that for normally distributed edge weights $E(T_n/(n(2 \ln n)^{1/2}) \to -1$ and that the weight of a greedy tour has the same behavior. Convergence in probability, which can be proved by the argument of Borovkov [1962], is a much stronger result, as we saw in Chapter 2. However, there seems to be hope that Lueker's arguments can be extended to prove stronger theorems about graph optimization problems for arbitrary edge weight distributions, much as we were able to extend Theorem 4.7 from the case of the TSP to a much wider class of problems.

As a final note, we observe that the argument used to prove Theorem 4.8 works even better in the case that we are solving a minimax optimization problem. Rather than seeking the feasible solution which minimizes the sum of the edge weights, we want to minimize the maximum edge weight. The analogue of the TSP in this setting is called the bottleneck TSP (see Garfinkel and Gilbert [1978]).

THEOREM 4.9 -

Let G_n be a random labelled graph in which each edge is independently present with probability p, and let C_n be a set of

labelled graphs on n vertices. Let p_n be such that if $p \geq p_n$ then G_n contains a member of C_n as a subgraph, almost surely in the independent model, and let q_n be such that if $p \leq q_n$ then G_n does not contain a member of C_n as a subgraph, almost surely. Given a complete graph on n vertices with edge weights chosen independently from the distribution F, let M_n be the minimum, over all graphs of C_n, of the maximum weight edge in such a graph. Then

$$F^{-1}(q_n^-) \leq M_n \leq F^{-1}(p_n)$$

almost surely in the independent model.

Proof - The upper bound follows from the same argument as we used in the proof of Theorem 4.8. For the lower bound, consider the subgraph of the complete weighted graph consisting of all n vertices and those edges of weight at most $F^{-1}(q_n^-)$. This is a random graph G_n with $p = F(F^{-1}(q_n^-)) \leq q_n$. Since this graph almost surely does not contain a member of C_n as a subgraph, the minimax solution must have an edge with weight greater than $F^{-1}(q_n^-)$. □

Theorem 4.9 provides not only an upper bound on the solution value, but a lower bound as well. From it, we can determine the behavior of the minimax solution values for several problems. For the spanning tree which minimizes the maximum edge weight, we can show that $M_n/F^{-1}(\ln n/n) \rightarrow_{as} 1$ in the independent model, assuming only

that F(x) is continuous for all sufficiently small x. For the bottleneck TSP, we have that $1 \leq M_n/F^{-1}(\ln n/n) \leq C$ almost surely, for some constant C, under the same condition on F. This improves on a result of Garfinkel and Gilbert [1978] describing bounds on the asymptotic behavior of $E(M_n)$ for the bottleneck TSP when edge weights are uniformly distributed.

4.2.2. Parallelism

With the advent of parallel computers, considerable attention has recently been given to designing and proving the correctness of parallel programs. For example, Kung and Song [1977] have given a proof of correctness for their parallel garbage collection algorithm, along with a detailed account of how it was designed. Baudet [1978] presents several techniques for designing algorithms for asynchronous multiprocessors.

Without this basis, the results of this section would seem purely theoretical. However, Baudet [1978] has analyzed the alpha-beta search algorithm and shown that it may exhibit behavior similar to that which we describe here. In particular, we will see that it is sometimes possible to achieve a speed-up of the average computation time for a problem by simulating parallelism on a single processor.

Our model of a multiprocessor is illustrated in Figure 4.1. There are k independent identical processors which share a common memory and which operate asynchronously. In short, each processor operates as though it were alone, except for the fact that its computations may be affected by other processors which change shared memory locations. We assume that there is no memory contention, and that

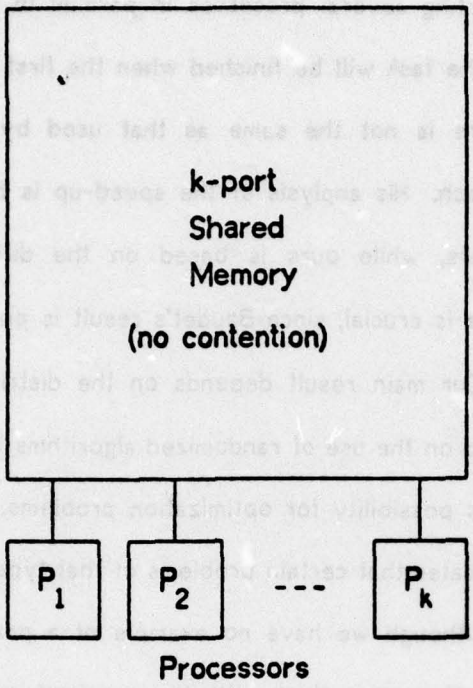


FIGURE 4.1 - An asynchronous multiprocessor.

each processor operates at the same speed regardless of the number of processors which may be running simultaneously.

A task which can be scheduled on such a machine is called a process. If there are more processors than processes active at any time, then each process is allocated one processor and the remaining processors are idle. If there are more processes than processors, then scheduling is by processor sharing, which means that each process effectively has only a fraction of a processor's computing power. If there are n processes active on k processors and if $n \geq k$, then each process effectively has k/n processors. We assume for the moment that there is no overhead associated with the scheduling policy, although this restriction will be removed in Section 4.2.3.

The main result of this section is that a speed-up can be achieved for certain

algorithms simply by starting several processes in parallel to do the same job, and relying on the fact that the task will be finished when the first process finishes. The mechanism operating here is not the same as that used by Baudet [1978], who suggests a similar approach. His analysis of the speed-up is based on the differing speeds of the processors, while ours is based on the differing speeds of the processes. The difference is crucial, since Baudet's result is purely an artifact of the model of computation. Our main result depends on the distribution of computation times for an algorithm and on the use of randomized algorithms (see Chapter 3). Diane Detig first suggested this possibility for optimization problems, and Baudet's analysis of alpha-beta search indicates that certain problems of that type might actually exhibit the desired behavior. Although we have no example of a practical algorithm which might have a distibution of running times with the required properties, the following lemma suggests that one is possible.

LEMMA 4.10 -

Let $F(x) = x^{\delta}$, $0 \le x \le 1$, for some $\delta > 0$. Let X be a random variable having the distribution F and $X_{1:k}$ be the smallest of k such independent random variables.

(a) $E(X) = \delta/(1+\delta)$.

(b) $E(kX_{1:k}) = k \cdot k! / \prod_{1 \le j \le k} (j + 1/\delta)$.

(c) If $\delta < 1/2$ then $E(kX_{1:k}) < E(X)$ for all $k \ge 2$.

(d) If $\delta < 1$ then $E(kX_{1:k}) < E(X)$ for all sufficiently large k.

Proof - Finding E(X) is very easy. Using Lemma 4.2 we see that $E(X_{1:k}) =$

$\sum_{0\le j\le k} \binom{k}{j}(-1)^j/(1+j\delta)$. Multiplying this expression by k and using a result of Knuth [1968] (section 1.2.6, exercise 48) proves part (b) of the lemma. By an extension of this argument, we can show that $E(kX_{1:k}) < E(X)$ for all $k \ge 2$ if $\delta < 1/2$, and that $E(kX_{1:k}) = o(1)$ if $\delta < 1$. □

Now suppose that F represents the distribution of solution times for a fixed instance of a problem using some randomized algorithm. The variation in solution times is due to the randomization, not variation in the problem inputs. If we ran the algorithm on that problem instance the expected computation time would be E(X). However, if we had a multiprocessor with k processors and ran the same algorithm on each one (so that there were k processes on the system), the expected solution time would be equal to the expected value of the minimum of k independent random variables chosen from F. The fact that the solution times are independent is a consequence of the randomization, since surely the problems being solved are not independent; in fact, they are all the same. The expected solution time would therefore be $E(X_{1:k})$. By application of Lemma 4.10, we see that if $F(x) = x^\delta$ for $\delta < 1/2$, the speed-up achieved in the average computation time would be $E(X)/E(X_{1:k}) > k$.

Equivalently, we could start up n processes on a single processor, using processor sharing, and the speedup would be $E(X)/E(nX_{1:n}) > 1$. The completion time for the simulated parallel version is n times the minimum of n i.i.d. random variables because each process effectively has 1/n of a processor.

Several points should be emphasized here. The first is simply that an algorithm

which has a distribution of running times satisfying the conditions of Lemma 4.10, part (d), does not have optimal expected running time, since we have explicitly demonstrated a method for improving it. In fact, if the behavior of F in the neighborhood of $x = 0$ is like that described in Lemma 4.10(d) then the expected running time is not optimal. This is a novel negative result which could be checked by experiments for any particular algorithm. On the other hand, we have no examples of algorithms which exhibit running times with such behavior, and it would be surprising if there were any algorithm for a practical problem having a running time distribution with the required high density near zero. Even if the distribution of computation times does not satisfy these conditions, though, it still may be possible to obtain a speed-up for certain values of n. Again, this could be easily checked in practice with a reasonable estimate of the execution time distribution.

Baudet [1978] has shown that the alpha-beta search algorithm might be improved by first dividing the range of solution values into two parts and solving the two subproblems independently. In much the same way as we propose to achieve a speed-up, Baudet simulates two processors by one and theoretically obtains an improved algorithm. The effect is valid only for two processes on a single processor, not for three or more, but suggests that a real speed-up may be possible using simulated parallelism.

No programming languages in common use today provide the facilities required for actual implementation of simulated parallelism on a single processor[2]. Those which

---

2 . There may be languages for real-time applications which explicitly permit processor sharing, but these are not readily available to users of a batch or time- sharing system.

allow parallelism, such as Algol 68, do not specify the scheduling policy for the parallel processes, without which the feature is useless to us. Occasionally, one even hears the claim that the absence of a scheduling policy is an advantage, because the programmer should not need to know how parallel tasks are scheduled in order to produce a correct program. While it may be true that correctness should not depend on the scheduling policy, we have seen that running time may depend on it, and have supplied a reason for choosing processor sharing in certain circumstances.

4.2.3. Problem Decomposition for Multiprocessors

In the previous section, we saw how it was possible, in theory at least, to improve a non-optimal algorithm by simulating parallelism. Here we examine the question of problem decomposition for a multiprocessor such as that of Figure 4.1 with the express purpose of making use of k processors to achieve a speed-up of k. There exists actual hardware, such as C.mmp (see Wulf and Bell [1972]), which is modelled reasonably well by the simple structure of Figure 4.1, so the problem to be considered is a real one.

Much previous work in parallel algorithms has centered on calculating the optimal speed-up for a given problem when there are an unlimited number of processors. With real multiprocessor systems such as C.mmp, the number of processors is not only fixed but is usually rather small. The C.mmp multiprocessor, for example, has k = 16. Since the optimal speed-up is k when there are k processors (for we could simulate the k processors by one using processor sharing) we will examine the question of how to obtain this optimal speed-up for a class of problems.

Assume that we have an algorithm for solving some problem on a single

processor, and wish to somehow decompose the problem into subproblems which could be solved independently on a multiprocessor. Let the problem and algorithm have the following properties:

(1) The time required by the algorithm to solve a random instance of the problem on a single processor is a random variable X having the distribution F(x). We assume that $F(x) = 0$ for $x < 0$ since processing times are non-negative, and that $\mu = E(X)$ is finite.

(2) The problem can be solved by solving all of any finite number n of subproblems, each of which is of the same type as the original problem but is smaller (in solution time) by a factor of n. Therefore, the solution time for each subproblem on a single processor is a random variable having the distribution F(nx).

(3) The subproblem solution times are independent of each other and independent of the solution time of the original problem.

Assumption (2) seems questionable at first glance, but is in fact quite reasonable, especially for many numerical linear algebra problems, discrete optimization problems, and queries in large data bases, for instance. Assumption (3) is unreasonable in most cases, but this is the price we must pay for the ability to get analytical results. Actually, the subproblem solution times may be almost independent if n is very large.

The problem to be considered is how to determine n such that the expected solution time on k processors is minimized. Each subproblem is allocated one process, and the total solution time is the elapsed time to completion of the last process which finishes, since all subproblems must be solved in order to solve the original problem.

Although in the model n can be arbitrarily large, presumably any real problem can be subdivided only so far before the assumptions fail. A conclusion such as "make n as large as possible" means "make n as large as possible such that the assumptions (1) through (3) above are satisfied".

Let $T_k(n)$ be the total solution time on k processors when the problem is divided into n subproblems; let $X_{j:n}$ be the j^{th} smallest of n random variables from the distribution F(x); and let $\mu_{j:n} = E(X_{j:n})$. Define $Y_{j:n}$ to be the solution time of the j^{th} subproblem if each subproblem (process) had its own processor. Then $E(Y_{j:n}) = \mu_{j:n}/n$ because of assumption (2). Finally, let S_j be the elapsed time to completion of the j^{th} subproblem using processor sharing on k processors.

In order to solve the problem, we must find an expression for $T_k(n)$. By definition, $T_k(n) = S_n$. Furthermore, note that $S_1 = (n/k)X_{1:n}$ (since there are n active processes before time S_1 and each has k/n effective processors) and that

$$S_j = S_{j-1} + (X_{j:n} - X_{j-1:n})(n-j+1) / k$$

for $2 \le j \le n-k$. This follows from the fact that between the completion of the $j-1^{st}$ and j^{th} subproblems, there are n-j+1 active processes. Iterating the recurrence we find that

$$S_{n-k} = X_{n-k:n} + (1/k) \sum_{1 \le j \le n-k} X_{j:n}.$$

After time S_{n-k} there are at most k processes still active, so each has its own processor and there is no sharing. The time remaining until all finish is simply $X_{n:n} - X_{n-k:n}$, so

$$T_k(n) = S_n = X_{n:n} + (1/k) \sum_{1 \le j \le n-k} X_{j:n} .$$

Taking expectations of both sides gives

$$E(T_k(n)) = \mu_{n:n}/n + (1/k) \sum_{1 \le j \le n-k} \mu_{j:n}/n$$

which can be rewritten as

$$E(T_k(n)) = \mu/k + \mu_{n:n}/n - (1/k) \sum_{n-k+1 \le j \le n} \mu_{j:n}/n .$$

This is the expression to be minimized. In order to do this, we will form $\Delta E(T_k(n)) = E(T_k(n)) - E(T_k(n-1))$, and then investigate its behavior for various values of $n > k$. This will allow us to determine for what values of n the expected total solution time is increasing and for which it is decreasing. If $\Delta E(T_k(n)) < 0$ then it is better to have n subproblems than n-1; otherwise, it is better to have only n-1. It is clear that it is always worthwhile to have at least k subproblems.

Forming the expression for $\Delta E(T_k(n))$, then applying Lemma 4.3 to get all variables in terms of expected values of order statistics from a sample of size n, and finally simplifying the sums gives the surprisingly simple expression

$$\Delta E(T_k(n)) = (\mu_{n-k:n} - \mu_{n-1:n}) / (n(n-1))$$

for $n > k$. Since $\mu_{n-k:n} \le \mu_{n-1:n}$ for any distribution, $\Delta E(T_k(n)) \le 0$ for all $n > k$. In fact, unless $k = 1$ or the distribution F is degenerate and all problems have identical solution times, $\Delta E(T_k(n)) < 0$, which means that we should make n as large as possible to minimize the expected total solution time. This proves

THEOREM 4.11 -

Under the assumptions (1) through (3) above, and assuming that there is no overhead associated with processor sharing, the total solution time is minimized if n is chosen to be as large as possible.

As users of real multiprocessor systems such as C.mmp well know, processor sharing is not implemented without overhead. Fortunately, it is possible to account for the overhead in a simple manner. An approximation to processor sharing is achieved by allowing each process to run for a small length of time (a "quantum") using round-robin scheduling of the active processes. The overhead is associated with "context swapping" from one process to the next. If we define c to be the ratio of the context swapping time to the quantum size, the total overhead incurred is cS_{n-k}, and

$$T_k(n) = S_n + cS_{n-k}$$

for $n > k$.

Proceeding in a fashion similar to that used above we find

$$\Delta E(T_k(n)) = \left(c\,(k+1)\,(\mu_{n-k:n} - \mu_{n-k-1:n}) + \mu_{n-k:n} - \mu_{n-1:n}\right) \Big/ (n(n-1))$$

where $\mu_{0:n} = 0$ by definition.

It is clear that we cannot make the same strong statement that we should always create as many subproblems as possible, regardless of c and F(x). Obviously, for large values of c we should avoid creating more processes than we have processors in order to avoid processor sharing. What is not so obvious is whether there is some optimal number of processes we should create for given c and F, and if so, what it is.

It turns out that it is helpful to define a new quantity $\delta_{j:n} = \mu_{n-j+1:n} - \mu_{n-j:n}$ which is the expected value of the difference between the j^{th} and $j+1^{st}$ largest (not smallest, as before) of n random variables from the distribution F. Writing $\Delta E(T_k(n))$ in terms of $\delta_{j:n}$, we have

$$\Delta E(T_k(n)) = \left(c\,(k+1)\,\delta_{k+1:n} - \sum_{2 \le j \le k} \delta_{j:n}\right) \Big/ (n(n-1))$$

where $\delta_{n:n} = \mu_{1:n}$. This means that

$$\Delta E(T_k(n)) < 0 \quad \text{iff} \quad c \le (1/(k+1)) \sum_{2 \le j \le k} (\delta_{j:n} / \delta_{k+1:n}) .$$

For certain forms of the distribution F there exist simple expressions for $\delta_{j:n}$, allowing us to compute $\Delta E(T_k(n))$ explicitly for certain cases. For example, for the exponential distribution $\delta_{j:n} = 1/j$ and for the uniform distribution $\delta_{j:n} = 1/(n+1)$. Therefore,

$$\Delta E(T_k(n)) < 0 \quad \text{iff} \quad c < H_k - 1 \qquad \text{(exponential)}$$

$$\Delta E(T_k(n)) < 0 \quad \text{iff} \quad c < (k-1)/(k+1) \qquad \text{(uniform)}$$

Since these conditions are independent of n, we may still reach the rather strong conclusion that if c satisfies the appropriate condition above than we should create as many subproblems as possible. If c is too large, then we should create exactly k subproblems. These conditions on c are extremely weak, since the value of c for a real system might be on the order of a few percent, while $c < 1/3$ suffices here even for only two processors.

For most other distributions, no such closed-form expressions for $\delta_{j:n}$ are

available. However, we can divide the possible values of n into three mutually exclusive and exhaustive ranges:

(1) $1 \le n \le k$: $\Delta E(T_k(n)) < 0$ always

(2) $n = k+1$: $\Delta E(T_k(n)) < 0$ iff $c < (\mu_{k:k+1} - \mu_{1:k+1}) / ((k+1)\,\mu_{1:k+1})$

(3) $n > k+1$: $\Delta E(T_k(n)) < 0$ iff $c < (1/(k+1)) \sum_{2 \le j \le k} (\delta_{j:n} / \delta_{k+1:n})$.

It is noteworthy that for $n > k+1$, the critical value of c (call it C) depends only on ratios of differences between expected values of order statistics. These ratios are, of course, distribution dependent, but are independent of location and scale parameters. Thus, if the distribution F is a gamma distribution, for instance, we can calculate the values of C for $n > k+1$ without knowledge of the actual mean and variance for F.

For the sake of argument, let us assume that the distribution of problem solution times is adequately represented by a gamma distribution. Tabulating the critical values for this distribution, we find that $c < 1/4$ is a sufficient condition for $\Delta E(T_k(n)) < 0$ for all $k \ge 2$ and $n > k+1$. Since c should be much smaller than that for a real system, we will assume that this condition is satisfied.

Now the only question is whether $\Delta E(T_k(k+1)) < 0$. If it is, then we should create as many subproblems as possible. If it is not, then we need to know whether the increase in solution time at $n = k+1$ can be offset by the known decreases thereafter. The second problem is easy, since $E(T_k(n)) \to \mu(1+c)/k$ as $n \to \infty$. Therefore, creating as many subproblems as possible is better than creating just k subproblems

iff $c < \mu_{k:k}/\mu - 1$. This apparently makes the answer to the former problem irrelevant, for if $\Delta E(T_k(k+1)) < 0$ we would certainly have $c < \mu_{k:k}/\mu - 1$. Hence, we have proved

THEOREM 4.12 -

Under the assumptions (1) through (3) above, if the distribution of solution times is a gamma distribution and if the ratio c of overhead to quantum size is at most 1/4, then the expected total solution time is minimized by letting $n = k$ whenever $c \geq \mu_{k:k}/\mu - 1$, and by making n as large as possible if $c < \mu_{k:k}/\mu - 1$.

The value of μ really only determines the time unit and may therefore be set to 1 without loss of generality. In this case, $\mu_{k:k} = V^2 z_{k:k}$, where $z_{k:k}$ is the expected value of the largest of k random variables from a gamma distribution with parameter $1/V^2$. The coefficient of variation of this distribution is V. In order to determine in practice how many subproblems to create we could estimate c and V from experimental data, look up $z_{k:k}$ in a table of expected values of order statistics (such as Pearson and Hartley [1972]), and decide on the basis of the criterion in Theorem 4.12.

4.3. Conclusions

While there have been many previous attempts to analyze the behavior of solutions to graph optimization problems, these have typically used ad hoc approaches which work only for one particular problem and for some simple distributions of the problem parameters. In this chapter we have shown how many of these results follow naturally from a few general theorems, and have demonstrated how the field of order

statistics provides the tools necessary to analyze probabilistic approximation algorithms for the more difficult optimization problems. The results obtained here are also stronger than the ad hoc results because they show convergence in probability or almost sure convergence of the random variables in question, as opposed to convergence of expected values.

In addition, tools from order statistics apply naturally to the analysis of algorithms for multiprocessors. This may prove to be more important in the future as more multiprocessing systems are implemented. Experience indicates that those who wish to study the behavior of parallel algorithms should be familiar with these concepts and with renewal theory as well as with the more traditional queuing theory.

5. References

Abramowitz, M. and Stegun, I.A., eds. Handbook of Mathematical Functions. Dover Publications, New York, 1965.

Baudet, G. The Design and Analysis of Algorithms for Asynchronous Multiprocessors. Ph.D. thesis, Carnegie-Mellon Univ., Dept. of Comp. Sci., Pittsburgh, PA, Apr. 1978.

Beardwood, J., Halton, J.H., and Hammersley, J.M. The shortest path through many points. Proc. Camb. Phil. Soc. 55, 4 (Oct. 1959), 299-327.

Bentley, J.L. Divide and Conquer Algorithms for Closest Point Problems in Multidimensional Space. Ph.D. thesis, Univ. of North Carolina, Dept. of Comp. Sci., Chapel Hill, NC, 1976.

Bentley, J.L., and Friedman, J.H. Algorithms and data structures for range queries. Proc. Comp. Sci. and Stat.: 11th Annual Symp. on the Interface. Inst. of Stat., N.C. State Univ., Raleigh, NC, Mar. 6-7, 1978, pp. 297-307.

Bentley, J.L., and Shamos, M.I. Divide and conquer for linear expected time. Inf. Proc. Lett. 7, 2 (Feb. 1978), 86-91.

Bentley, J.L., Stanat, D.F., and Williams, E.H. The complexity of finding fixed-radius near neighbors. Inf. Proc. Lett. 6, 6 (Dec. 1977), 209-212.

Bentley, J.L., Weide, B.W., and Yao, A.C. Optimal expected time algorithms for closest

point problems. To appear in Proc. 16th Annual Allerton Conf. on Comm., Contr., and Computing, 1978.

Blum, M., Floyd, R.W., Pratt, V., Rivest, R.L., and Tarjan, R.E. Time bounds for selection. J. Comput. Sys. Sci. 7, 4 (Aug. 1973), 448-461.

Borovkov, A.A. A probabilistic formulation of two economic problems. Dokl. Akad. Nauk SSSR 146, 5 (1962), 983-986; English translation in Sov. Math. Dokl. 3, 5 (1962), 1403-1406.

Burtin, Yu.D. On extreme metric parameters of a random graph, I: asymptotic estimates. Theory of Prob. Appl. 19, 4 (1974), 710-725.

Burtin, Yu.D. On extreme metric characteristics of a random graph, II: limit distributions. Theory of Prob. Appl. 20, 1 (1975), 83-101.

Carter, J.L., and Wegman, M.N. Universal classes of hash functions. Proc. 9th Annual ACM Symp. on Theory of Comput. ACM, New York, 1977, pp. 106-112.

Chung, K.L. A Course in Probability Theory, 2nd ed. Academic Press, New York, 1974.

Christofides, N. Worst-case analysis of a new heuristic for the travelling salesman problem. Algorithms and Complexity: New Directions and Recent Results, J.F. Traub, ed. Academic Press, New York, 1976, p. 441.

Croes, A. A method for solving traveling-salesman problems. Op. Res. 6, 6 (Nov.-Dec. 1958), 791-812.

David, H.A. Order Statistics. John Wiley and Sons, Inc., New York, 1970.

DeWitt, H.K. The Theory of Random Graphs with Applications to the Probabilistic Analysis of Optimization Algorithms. Ph.D. thesis, Univ. of California, Dept. of Comp. Sci., Los Angeles, CA, 1977.

Dobosiewicz, W. Sorting by distributive partitioning. Inf. Proc. Lett. 7, 1 (Jan. 1978), 1-6.

Erdos, P., and Renyi, A. On random graphs, I. Publ. Math. 6, (1959), 296-297.

Erdos, P., and Spencer, J. Probabilistic Methods in Combinatorics. Academic Press, New York, 1974.

Feller, W. An Introduction to Probability Theory and Its Applications, Vol. I. John Wiley and Sons, Inc., New York, 1968.

Floyd, R.W., and Rivest, R.L. Expected time bounds for selection. Comm. ACM 18, 3 (Mar. 1975), 165-173.

Garey, M.R., Graham, R.L., and Johnson, D.S. Some NP-complete geometric problems. Proc. 8th Annual ACM Symp. on Theory of Comput. ACM, New York, 1976, pp. 10-22.

Garey, M.R., and Johnson, D.S. Approximation algorithms for combinatorial problems: an annotated bibliography. Algorithms and Complexity: New Directions and Recent Results, J.F. Traub, ed. Academic Press, New York, 1976, pp. 41-52.

Garfinkel, R.S., and Gilbert, K.C. The bottleneck traveling salesman problem: algorithms and probabilistic analysis. J. ACM 25, 3 (July 1978), 435-448.

Garfinkel, R.S., and Nemhauser, G.L. Integer Programming. Wiley and Sons, Inc., New York, 1972.

Gill, J.T. Computational complexity of probabilistic Turing machines. Proc. 6th Annual ACM Symp. on Theory of Comput. ACM, New York, 1974, pp. 91-95.

Gimady, E.Kh., Glebov, N.I., and Perepelica, V.A. Approximation algorithms for discrete optimization problems. Problemy Kibernetiki 31, (1976), 35-42; English translation by Michael Shamos.

Golden, B.L. A statistical approach to the TSP. Networks 7, (Fall 1977), 209-225.

Graham, R.L. Bounds for certain multiprocessing anomalies. Bell Syst. Tech. J. 45, (1966), 1563-1581.

Graham, R.L. An efficient algorithm for determining the convex hull of a finite planar set. Inf. Proc Lett. 1, (1972), 132-133.

Grimmett, G.R., and McDiarmid, C.J.H. On colouring random graphs. Math. Proc. Camb. Phil. Soc. 77, (1975), 313-324.

Gumbel, E.J. Statistics of Extremes. Columbia University Press, New York, 1958.

Halmos, P.R. Measure Theory. D. Van Nostrand Co., Inc., Princeton, NJ, 1950.

Harary, F. Graph Theory. Addison-Wesley, Reading, MA, 1969.

Hardy, G.H. Orders of Infinity, 2nd ed. Cambridge Univ. Press, London, 1924.

Hartigan, J.A. Necessary and sufficient conditions for asymptotic joint normality of a statistic and its subsample values. Annals of Stat. 3, 3 (1975), 573-580.

Held, M., and Karp, R.M. A dynamic programming approach to sequencing problems. J. SIAM 10, (1962), 196-210.

Hoare, C.A.R. Quicksort. Comput. J. 5, (1962), 10-15; also, Algorithm 64: quicksort. Comm. ACM 4, (Apr. 1961), 321.

Holland, J.H. Adaptation in Natural and Artificial Systems. Univ. of Michigan Press, Ann Arbor, 1975.

Horowitz, E., and Sahni, S. Combinatorial Problems: Reducibility and Approximation. TR 77-14, Univ. of Minnesota, Comp. Sci. Dept., Minneapolis, MN, Sept. 1977.

Karp, R.M. The probabilistic analysis of some combinatorial search algorithms. Algorithms and Complexity: New Directions and Recent Results, J.F. Traub, ed. Academic Press, New York, 1976, pp. 1-19.

Karp, R.M. Probabilistic analysis of partitioning algorithms for the traveling-salesman problem in the plane. Math. of Op. Res. 2, 3 (Aug. 1977), 209-224.

Kernighan, B.W., and Lin, S. An efficient heuristic procedure for partitioning graphs. Bell Syst. Tech. J. 49, 2 (Feb. 1970), 291-308.

Kernighan, B.W., and Lin, S. Heuristic solution of a signal design problem. Bell Syst. Tech. J. 52, 7 (Sept. 1973), 1145-1159.

Knuth, D.E. The Art of Computer Programming, Vol. I: Fundamental Algorithms. Addison-Wesley, Reading, MA, 1968.

Knuth, D.E. The Art of Computer Programming, Vol. II: Seminumerical Algorithms. Addison-Wesley, Reading, MA, 1969.

Knuth, D.E. The Art of Computer Programming, Vol. III: Sorting and Searching. Addison-Wesley, Reading, MA, 1973.

Knuth, D.E. Big omicron and big omega and big theta. SIGACT News 8, 2(Apr.-June 1976), 18-24.

Kung, H.T., and Song, S.W. An efficient parallel garbage collection system and its correctness proof. Proc. 18th Annual IEEE Symp. on Found. of Comput. Sci. IEEE, New York, 1977, pp. 120-131.

Lewis, H.R., and Papadimitriou, C.H. The efficiency of algorithms. Sci. Am. 238, 1 (Jan. 1978), 96-109.

Lin, S. Computer solutions of the traveling salesman problem. Bell Syst. Tech. J. 44, 10 (Dec. 1965), 2245-2269.

Lin, S. Heuristic programming as an aid to network design. Networks 5, 1 (Jan. 1975), 33-43.

Lin, S., and Kernighan, B.W. An effective heuristic algorithm for the traveling-salesman problem. Op. Res. 21, 2 (Mar.-Apr. 1973), 498-516.

Lipton, R.J., and Tarjan, R.E. Applications of a planar separator theorem. Proc. 18th Annual IEEE Symp. on Found. of Comput. Sci. IEEE, New York, 1977, pp. 162-170.

Lueker, G.S. Maximization problems on graphs with edge weights chosen from a normal distribution. Proc. 10th Annual ACM Symp. on Theory of Comp., ACM, New York, 1978, pp. 13-18.

Nishizeki, T., Hidenao, O., and Saito, N. Computational complexity of k-maximal cuts. Proc. 15th Annual Allerton Conf. on Comm., Contr., and Computing. Univ. of Ill., Urbana-Champaign, Ill., Sept. 28-30, 1977, pp. 567-576.

Papadimitriou, C.H. [1977a] The Euclidean traveling salesman problem is NP-complete. Theoret. Comput. Sci. 4, 3 (1977), 237-244.

Papadimitriou, C.H. [1977b] The probabilistic analysis of matching heuristics. Proc. 15th Annual Allerton Conf. on Comm., Contr., and Computing. Univ. of Ill., Urbana-Champaign, Ill., Sept. 28-30, 1977, pp. 368-378.

Pearson, E.S., and Hartley, H.O., eds. Biometrika Tables for Statisticians, Volume II. Cambridge University Press, London, 1972.

Perepelica, V.A. On two problems from the theory of graphs. Dokl. Akad. Nauk SSSR Tom 194, 6 (1970), 1269-1272; English translation in Sov. Math. Dokl. 11, 5 (1970), 1376-1379.

Pickands, J. Moment convergence of sample extremes. Ann. Math. Stat. 39, 3 (1968), 881-889.

Posa, L. Hamiltonian circuits in random graphs. Discrete Math. 14, (1976), 359-364.

Rabin, M.O. Probabilistic algorithms. Algorithms and Complexity: New Directions and Recent Results, J.F. Traub, ed. Academic Press, New York, 1976, pp. 21-39.

Robinson, J.T. Some analysis techniques for asynchronous multiprocessor algorithms. IEEE Trans. on Software Eng., to appear in 1978.

Sedgewick, R. Quicksort. Ph.D. thesis, Stanford Univ., Comp. Sci. Dept., Stanford, CA,

May 1975; for a summary see Sedgewick, R. The analysis of quicksort programs. Acta Inf. 7, 4 (1977), 327-355.

Sedgewick, R. Permutation generation methods. Comput. Surv. 9, 2 (June 1977), 137-164.

Sen, P.K. On the moments of the sample quantiles. Calcutta Stat. Assoc. Bullet. 9, 33 (Sept. 1959), 1-19.

Shamos, M.I. Geometric complexity. Proc. 7th Annual ACM Symp. on Theory of Comp., ACM, New York, 1975, pp. 224-233.

Shamos, M.I. Geometry and statistics: problems at the interface. Algorithms and Complexity: New Directions and Recent Results, J.F. Traub, ed. Academic Press, New York, 1976, pp. 251-280.

Shamos, M.I. Computational Geometry. Ph.D. thesis, Yale Univ.; available from Carnegie-Mellon Univ., Dept. of Comp. Sci., Pittsburgh, PA, 1978.

Shamos, M.I., and Hoey, D. Closest point problems. Proc. 16th Annual IEEE Symp. on Found. of Comput. Sci. IEEE, New York, 1975, pp. 151-162.

Shamos, M.I., and Yuval, G. Lower bounds from complex function theory. Proc. 17th Annual IEEE Symp. on Found. of Comput. Sci. IEEE, New York, 1976, pp. 268-273.

Solovay, R., and Strassen, V. A fast Monte-Carlo test for primality. SIAM J. Comput. 6, 1 (Mar. 1977), 84-85.

Spira, P.M. A new algorithms for finding all shortest paths in a graph of positive arcs in average time $O(n^2 \log^2 n)$. SIAM J. Comput. 2, 1 (Mar. 1973), 28-32.

Stein, D.M. An asymptotic, probabilistic analysis of a routing problem. Math. of Op. Res. 3, 2 (May 1978). 89-101.

Tukey, J.W. The ninther, a technique for low-effort robust (resistant) location in large samples. Contributions to Survey Sampling and Applied Statistics, H.A. David, ed. Academic Press, New York, 1978, pp. 251-257.

Vajda, S. Probabilistic Programming. Academic Press, New York, 1972.

Walker, A.M. On the asymptotic distribution of sample quantiles. J. Royal Stat. Soc. 30, 3 (1968), 570-575.

Weide, B.W. Analysis of a Model of Problem Decomposition for Asynchronous Multiprocessors. Unpublished technical memo, Carnegie-Mellon Univ., Dept. of Comp. Sci., Pittsburgh, PA, May 1976.

Weide, B.W. A survey of analysis techniques for discrete algorithms. Comp. Surv. 9, 4 (Dec. 1977), 291-313.

Weide, B.W. Space-efficient on-line selection algorithms. Proc. Comp. Sci. and Stat.: 11th Annual Symp. on the Interface. Inst. of Stat., N.C. State Univ., Raleigh, NC, Mar. 6-7, 1978, pp. 308-312.

Wulf, W.A., and Bell, C.G. C.mmp - a multi-mini-processor. Proc. FJCC '72. 1972, pp. 765-777.

Yao, A.C. Probabilistic computations: toward a unified measure of complexity. Proc. 18th Annual IEEE Symp. on Found. of Comput. Sci. IEEE, New York, 1977, pp. 222-227.

Yuval, G. [1975a] Finding near neighbors in k-dimensional space. Inf. Proc. Lett. 3, 4 (Mar. 1975), 113-114.

Yuval, G. [1975b] personal communication, 1975.

Yuval, G. Finding nearest neighbors. Inf. Proc. Lett. 5, 3 (Aug. 1976), 63-65.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER CMU-CS-78-142	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) STATISTICAL METHODS IN ALGORITHM DESIGN AND ANALYSIS		5. TYPE OF REPORT & PERIOD COVERED Interim 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Bruce W. Weide		8. CONTRACT OR GRANT NUMBER(s) N00014-76-C-0370
9. PERFORMING ORGANIZATION NAME AND ADDRESS Carnegie-Mellon University Computer Science Dept Schenley Park, PA 15213		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research Arlington, VA 22217		12. REPORT DATE August 1978 13. NUMBER OF PAGES 184
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Same as above		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The use of statistical methods in the design and analysis of discrete algorithms is explored. Among the design tools are randomization, ranking, sampling and subsampling, density estimation, and "cell" or "bucket" techniques. The analysis techniques include those based on the design methods as well as the use of stochastic convergence concepts and order statistics. → next page

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

cont.

The introductory chapter contains a literature survey and background material on probability theory. In Chapter 2, probabilistic approximation algorithms are discussed with the goal of exposing and correcting some oversights in previous work. Some advantages of the proposed solution to the problems encountered are the introduction of a model for dealing with random problems, and a set of methods for analyzing the probabilistic behavior of approximation algorithms which permit consideration of fairly complex algorithms in which there are dependencies among the random variables in question.

Chapter 3 contains many useful design and analysis tools such as those mentioned above, and several examples of the uses of the methods. Algorithms which run in linear expected time for a wide range of probabilistic assumptions about their inputs are given for problems ranging from sorting to finding all nearest neighbors in a point set in k dimensions. Empirical results are presented which indicate that the sorting algorithm, Binsort, is a good alternative to Quicksort under most conditions. There are also new algorithms for some selection and discrete optimization problems.

Finally, Chapter 4 describes the uses of results from order statistics to analyze greedy algorithms and to investigate the behavior of parallel algorithms. Among the results reported here are general theorems regarding the distribution of solution values for optimization problems on weighted graphs. Many recent results in the literature, which apply for certain distributions of edge weights and for specific problems, follow as immediate corollaries from these general theorems.